

LATEX for Administrative Work

Version 1.1

Nicola L. C. Talbot

Dickimaw Books
www.dickimaw-books.com

30th September, 2015

Copyright © 2015 Nicola L. C. Talbot

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and one Back-Cover Text: "If you choose to buy a copy of this book, Dickimaw Books asks for your support through buying the Dickimaw Books edition to help cover costs." A copy of the license is included in the section entitled "[GNU Free Documentation License](#)".

The base URL for this document is: [http://www.dickimaw-books.com/
latex/admin/](http://www.dickimaw-books.com/latex/admin/)

CONTENTS

LIST OF EXERCISES	xviii
LIST OF EXAMPLES	xx
1 INTRODUCTION	1
1.1 Packages and Document Classes	6
1.2 Arara	12
2 MANAGING DATA	21
2.1 Utility Commands	23
2.1.1 Macro Definitions	24
2.1.2 Hook Management	48
2.1.3 Arithmetic	55

Contents

2.2	■ Loading Data	69
2.2.1	■ Loading Data From a CSV File	70
2.2.2	■ Loading Data From a .dbtex File	82
2.3	■ Security	104
2.4	■ Sorting Data	110
2.5	■ Sample Data	115
2.5.1	■ Sample CSV Files	117
2.5.2	■ Sample XLS File	123
2.5.3	■ Sample SQL Tables	126
2.6	■ Displaying Tabulated Data	137
2.7	■ Iteration	149
2.7.1	■ Iterating Through a Database	149
2.7.2	■ Iterating Over a Comma-Separated List	156
2.7.3	■ Iteration With etoolbox's Internal Lists	172
2.7.4	■ General Iteration with TeX's \loop	179
2.7.5	■ Iteration Tips and Tricks	180
2.8	■ Fetching Data From a Given Row	196

Contents

2.9	■ Null and Boolean Values	209
3	■ CORRESPONDENCE	219
3.1	■ Writing a Letter Using the letter Class	220
3.2	■ Writing a Letter Using the scrletter2 Class	227
3.3	■ Writing a Letter Using the newlfm Class	249
3.4	■ Writing a Letter Using the isodoc Class	260
3.5	■ Mail Merging	268
3.6	■ Envelopes	278
4	■ INVOICES	305
4.1	■ Writing an Invoice Using the isodoc Class	306
4.2	■ Writing an Invoice Using the invoice Package	313
4.3	■ Building Your Own Invoice using longtable and datatool . .	321
5	■ CURRICULA VITÆ (RÉSUMÉS)	334
5.1	■ The currvita Package	335

Contents

5.2	■ The europecv Class	340
5.2.1	■ Setting Personal Information	343
5.2.2	■ Sections and Publication Lists	349
5.2.3	■ Spoken Languages	368
6	■ OFFICIAL DOCUMENTS	374
6.1	■ Memos	374
6.2	■ Press Releases	378
6.3	■ Minutes	387
6.3.1	■ The meetingmins Class	388
6.3.2	■ The minutes Package	394
6.4	■ Confidentiality	410
6.4.1	■ Redaction: The censor Package	410
6.4.2	■ Watermarks	422
6.5	■ Typesetting Legal Documents (Numbered Paragraphs)	434
6.5.1	■ Hierarchical Paragraph Numbering	436
6.5.2	■ Non-Hierarchical Paragraph Numbering	450

Contents

7	 DATES AND TIMES	460
7.1	 The datetime2 Package	462
7.2	 The pgfcalendar Package Utility Commands	468
7.3	 Displaying a Date	481
7.4	 Parsing and Displaying Times	499
7.5	 Displaying a Calendar	521
8	 PRESENTATIONS (THE beamer CLASS)	570
8.1	 Overlays	581
8.2	 Themes	590
9	 ASSIGNMENTS AND EXAMINATIONS	606
9.1	 The exam Class	608
9.2	 The exsheets Package	626
9.3	 The probsoln Package	638
9.4	 Using the datatool Package for Exams or Assignment Sheets	671
9.5	 Random Numbers	700

Contents

10	BUSINESS CARDS, FLYERS AND LEAFLETS	708
10.1	The picture Environment	710
10.2	The ticket Package	722
10.3	The leaflet Class	730
10.4	The pst-barcode Package	748
10.5	The flowfram Package and the flowframtk Application	758
11	FORMS	810
11.1	Writing a Class File for a Form	819
11.2	Electronic PDF Forms	871
12	CHARTS	896
12.1	Flow Charts	898
12.2	Pie Charts	911
12.2.1	The datapie Package	913
12.2.2	The pgf-pie Package	927
12.3	Bar Charts	932
12.3.1	The bchart Package	934

Contents

12.4	■ Gantt Charts	948
12.5	■ Plots	958
13	■ COLLABORATING ON DOCUMENTS	974
13.1	■ Change Markup	981
13.2	■ Version Control	988
13.3	■ Online L ^A T _E X Editors	1003
ACKNOWLEDGEMENTS		1033
BIBLIOGRAPHY		1034
GLOSSARY		1050
SUMMARY OF COMMANDS AND ENVIRONMENTS		1058
INDEX		1203
GNU FREE DOCUMENTATION LICENSE		1244

Contents

HISTORY	1260
----------------	-------------

LIST OF FIGURES

2.1	datatoolk in Graphical Mode	88
2.2	Import CSV File	89
2.3	Imported CSV Data Shown in Main Window	90
2.4	Import SQL Data	95
2.5	Imported SQL Data Shown in Main Window	96
2.6	Changing the Column Header Text	97
2.7	Running datatoolk via arara	103
2.8	Sample XLS File (First Sheet)	125
2.9	Processing \expandafter	160
3.1	A Simple Letter Using the letter Class	225
3.2	A Simple Letter Using the scrlltr2 Class	231
3.3	A Simple Letter Using KOMA-Script Variables	248
3.4	A Simple Letter Using the newlfm Class	259
3.5	A Simple Letter Using the isodoc Class	269
3.6	Address Labels	280
3.7	Custom Big Label	287

List of Figures

3.8	Custom Formatted Big Label	290
4.1	Invoice Using the <code>isodoc</code> Class	311
4.2	Sample Invoice (<code>invoice</code> package)	318
4.3	Final Page of <code>longtable</code> Displaying Countries	327
5.1	Personal Information Section (<code>europecv</code> class)	348
5.2	Curriculum Vitæ Sections (<code>europecv</code> class)	352
5.3	A Tabulated Bibliography (First Page)	362
5.4	Example Language Table	372
5.5	Example Language Table (<code>booktabs</code> option)	373
6.1	A Sample Memo (<code>newlfm</code> class)	379
6.2	A Sample Press Release (<code>newlfm</code> class)	383
6.3	Sample Minutes (Title Information Only)	400
6.4	Sample Watermark (background package)	427
6.5	Sample Watermark (<code>xwatermark</code> package)	435
7.1	Calendar (Days of March)	531
7.2	Calendar (Node Shapes Added)	535
7.3	Calendar (Minimum Width Set on Nodes)	538

List of Figures

7.4	Calendar Using Circular and Rectangular Nodes (March 2014)	541
7.5	Calendar with Split Nodes (May 2014)	550
7.6	May 2014 with Bank Holidays	554
7.7	May 2014 with Bank Holidays (including end of previous month and beginning of next month)	563
8.1	Title Frame	574
8.2	An Example Frame	576
8.3	Verbatim in a Frame	578
8.4	Block	582
8.5	Overlays (First Slide)	584
8.6	Overlays (Second Slide)	585
8.7	Overlays (transparent option)	587
8.8	Boadilla Theme (Title Slide)	595
8.9	Boadilla Theme	596
8.10	EastLansing Theme (Title Slide)	597
8.11	EastLansing Theme	598
8.12	Montpellier Theme (Title Slide)	599
8.13	Montpellier Theme	600
8.14	Goettingen Theme (Title Slide)	601
8.15	Goettingen Theme	602

List of Figures

8.16	Goettingen with rounded and spruce Themes (Title Slide)	604
8.17	Goettingen with rounded and spruce Themes	605
9.1	Importing a <code>probsoln</code> Dataset into <code>datatooltk</code>	672
9.2	New Level Column Added	674
9.3	The <code>mth102</code> Database	699
9.4	Random Selection with <code>pgfmath</code> and <code>probsoln</code>	707
10.1	A Postcard	723
10.2	Name Labels (<code>ticket</code> package)	731
10.3	A Sample Leaflet (First Sheet)	744
10.4	A Sample Leaflet (Second Sheet)	745
10.5	A Sample Leaflet Using <code>tikz</code> (First Sheet)	749
10.6	Overlapping Frames (<code>flowfram</code> package)	766
10.7	Advance Information Sheet	773
10.8	FlowframTk Main Window	775
10.9	FlowframTk — Set the Storage Unit	776
10.10	FlowframTk — Set the Grid	778
10.11	FlowframTk — Set the $\text{\TeX}/\text{\LaTeX}$ Settings	779
10.12	FlowframTk — Setting the Preamble	781
10.13	FlowframTk — Setting the Margins	783
10.14	FlowframTk — Create a Rectangle	784

List of Figures

10.15	FlowframTk — Setting the Fill Colour	786
10.16	FlowframTk — Setting Flow Frame Data	787
10.17	FlowframTk — Flow Frame Data Assigned	788
10.18	FlowframTk — Setting Static Frame Data	790
10.19	FlowframTk — Saving the Image	791
10.20	FlowframTk — Assigning Dynamic Frame Data	792
10.21	FlowframTk — Setting the Frame Contents	794
10.22	FlowframTk — Bitmap Options	795
10.23	FlowframTk — Bitmap Selector	797
10.24	FlowframTk — Scaling	798
10.25	FlowframTk — Bitmap Moved to the Right	799
10.26	FlowframTk — Logo Added	801
10.27	FlowframTk — Setting the bookdata Frame	803
10.28	FlowframTk — Data Completed	804
10.29	FlowframTk — Export to a Class File	805
10.30	Advance Information Sheet (via flowframtk)	809
11.1	A Simple Form with Two Fill-In Areas and Two Check Boxes	838
11.2	A Simple Form with Multiple Check Box Areas	862
11.3	A Simple Form with Multiple Check Box Areas and a Fill-In Other Area	863
11.4	A Simple Form with a Text Area	872

List of Figures

11.5	Example Interactive PDF Form Viewed in Adobe Reader	880
11.6	Example Interactive PDF Form Viewed in Google Chrome	881
11.7	Example Interactive PDF Form Viewed in Okular	882
11.8	Example Interactive PDF Form Viewed in Sumatra	883
11.9	Example Interactive PDF Form Viewed in Foxit on Linux	884
11.10	Example Interactive PDF Form Viewed in Foxit on Windows	885
11.11	First Attempt at Laying Out Check Boxes in Rows and Columns	894
11.12	Second Attempt at Laying Out Check Boxes in Rows and Columns	895
11.13	Third Attempt at Laying Out Check Boxes in Rows and Columns	895
12.1	Nodes Positioned Relative to Each Other	902
12.2	Node Shapes Drawn and Filled	904
12.3	Nodes with Connecting Arrows	908
12.4	An Example Flow Chart	912
12.5	An Example Pie Chart (datapie package)	920
12.6	An Example Pie Chart with a Legend (datapie package)	925
12.7	An Example Pie Chart (pgf-pie package)	933
12.8	A Bar Chart (bchart package)	938
12.9	Bar Chart (bchart package) Exercise	939

List of Figures

12.10 A Bar Chart (databar package)	949
12.11 A Gantt Chart	959
12.12 A Plot (First Attempt)	968
12.13 A Plot (Second Attempt)	972
12.14 A Plot (Final Attempt)	973
13.1 Recording Changes	989
13.2 Subversion Revision Information	998
13.3 Overleaf (Mozilla Firefox)	1013
13.4 Overleaf—Rich Text Enabled	1015
13.5 Overleaf—Project Files	1017
13.6 Share ^L T _E X (Mozilla Firefox)	1019
13.7 Share ^L T _E X—Settings	1021
13.8 Overleaf—Error Handling	1022
13.9 Share ^L T _E X—Error Handling	1024
13.10 Overleaf—Comment Viewed as Rich Text	1028

LIST OF TABLES

2.1	Special Character Mappings	74
2.2	Products	140
2.3	Data Imported From shop.xls	146
2.4	Formatted Data Imported From shop.xls	154
2.5	Hardback Books	155
2.6	Paperback Books	155
2.7	Ebooks	155
2.8	Lists for Iteration Exercise	171
2.9	Customers (CSV)	217
2.10	Customers (SQL)	217
2.11	Customers (Check for Null or Empty and Boolean)	218
3.1	Common scrhtt2 Variables	236
3.2	Common scrhtt2 Options with Some of their Values	237
3.3	Letter Options for the newlfm Class	255
3.3	Letter Options for the newlfm Class (Continued)	256
3.4	Envelope Options for the envelab Package	282

List of Tables

3.5 Standard Label Options for the <code>envlab</code> Package	283
3.6 Big Label Options for the <code>envlab</code> Package	284
5.1 Convenient Shortcut Commands for <code>\ecvCEF</code>	370
6.1 Memo Options	376
6.2 A Redacted Table	413
6.3 Starred Redaction Commands: Project Success Rates . . .	417
6.4 Starred Redaction Commands (No Censoring): Project Suc- cess Rates	418
7.1 Ages	480
13.1 Share ^L AT _E X Plans	1004
13.2 Overleaf Plans	1007
13.2 Overleaf Plans (Continued)	1008
13.2 Overleaf Plans (Continued)	1009

LIST OF EXERCISES

1	Loading and Displaying Data	147
2	Iterating Through Data	156
3	Iterating Through a List	171
4	Internal Lists	178
5	Oxford Comma	194
6	Fetching a Row of Data	208
7	Writing a Letter (letter class)	226
8	Writing a Letter (scrlttr2 class)	247
9	Writing a Letter (newlfm class)	258
10	Mail Merging	278
11	Mail Merging With Envelope Labels Using newlfm, envlab and datatool	304
12	Creating an Invoice for a Customer (isodoc class)	312
13	Creating an Invoice for a Customer (invoice package)	319
14	Custom Invoice	333
15	Sample CV with Publications	340
16	List of Selected Publications (europecv class)	367

List of Exercises

17	Press Release (pressrelease class)	387
18	Minutes and Agendas (meetingmins class)	394
19	Minutes (minutes package)	409
20	Extending the Maximum enumerate Depth	448
21	Numbered Paragraphs	457
22	Displaying Times	519
23	Calendar for 2014	569
24	Creating an Exam Paper with the exam Class	625
25	Creating an Exam Paper with the exsheets Package	638
26	Creating an Assignment Sheet with the datatool Package	697
27	Name Labels (Iteration)	730
28	A QR Code	758
29	Query Form	819
30	Simple Form Class with Check Boxes	859
31	A Pie Chart (datapie package)	924
32	A Bar Chart (bchart package)	939

LIST OF EXAMPLES

1	Convert a .csv File to a .dbtex File	85
2	Convert an .xls Sheet to a .dbtex File	91
3	Importing SQL Data to a .dbtex File	93
4	Display Product List	139
5	Displaying Data Imported from a Spreadsheet	145
6	Iterating Through Data	152
7	List of Names	186
8	Fetching the Data From Row 1	197
9	Fetching a Customer's Details	200
10	Fetching a Customer's Details (With Expansion)	203
11	Display Customer List (Null Values)	213
12	Writing a Simple Letter (letter class)	222
13	A Simple Letter (scrltr2 class)	229
14	Writing a Letter: KOMA Settings	245
15	A Simple Letter (newlfm class)	256
16	A Simple Letter (isodoc class)	266
17	Mail Merging (letter class)	270

List of Examples

18	Mail Merging (newlfm class)	273
19	Envelope Labels	284
20	Mail Merging with letter and envlab	294
21	Mail Merging with newlfm, envlab and datatool	298
22	An Invoice (isodoc class)	309
23	An Invoice (invoice package)	317
24	Multi-Paged Tabulated Material	324
25	A Sample CV	337
26	Personal Information Section (europecv class)	346
27	Curriculum Vitæ With Sections (europecv class)	349
28	Tabulating a Bibliography	360
29	List of Publications (europecv class)	364
30	Sample Memo (newlfm class)	377
31	Press Release (newlfm class)	380
32	Sample Minutes (minutes package)	397
33	Redaction	420
34	Watermarks (background package)	425
35	Watermarks (xwatermark package)	433
36	Nested Enumeration	439
37	Displaying Dates and Times (datetime2 package)	467
38	Calculating Ages	477
39	Custom Date Formatting	487

List of Examples

40	Custom Date Package	494
41	Custom Date and Time Package	509
42	Calendar for May 2014	547
43	Presentation Themes (Boadilla)	592
44	Using Both the <code>exam</code> Class and the <code>probsoln</code> Package	646
45	Randomly Selecting Problems	667
46	Creating a Problem Sheet using <code>datatool</code>	676
47	Randomly Selecting Problems Not Used in the Past Two Years	688
48	Random Selection with <code>pgfmath</code> and <code>probsoln</code>	704
49	A Postcard	720
50	Name Labels (<code>ticket</code> package)	728
51	Sample Leaflet	735
52	Leaflet (with <code>tikz</code>)	746
53	ISBN Bar Code	754
54	Advance Information Sheet	767
55	Advance Information Sheet (with <code>flowframtk</code>)	774
56	A Simple Form Class	835
57	A Simple Form Class (Gathering Environment Contents) . .	868
58	A Simple Electronic Form	876
59	Flow Chart	907
60	An Example Pie Chart (<code>datapie</code> package)	918
61	A Pie Chart (<code>pgf-pie</code> package)	931

List of Examples

62	An Example Bar Chart (databar package)	946
63	A Gantt Chart	956
64	A Sample Plot (pgfplots package)	965
65	Recording Changes (changes package)	986
66	Accessing Subversion Revision Information	992

1. INTRODUCTION

This book is aimed at people who want to use L^AT_EX for administrative work, such as writing correspondence, performing repetitive tasks or typesetting problem sheets or exam papers. If you have never used L^AT_EX before, I recommend that you first read Volume 1: *L^AT_EX for Complete Novices* [93], since this book assumes you are already familiar with L^AT_EX.

As with all the books in this series, this tutorial only shows the basic usage of class files and packages. For more advanced commands, you will need to consult the class or package documentation (see Volume 1 [93, §1.1]). The reason for this is that it would be far too overwhelming for most readers to be presented with every possible option. (Consider, for example, the KOMA-Script manual is over 300 pages and the datatool user guide is over 200 pages at the time of writing.)

Throughout this document there are pointers to related topics in the [UK List of TeX Frequently Asked Questions \(UK FAQ\)](#). These are displayed in the margin in square brackets, as illustrated on the right. You may find these resources useful in answering related questions that are not covered in this book.

[FAQ: What is
LaTeX?]

On-line versions of this book, along with associated files and solutions to the exercises, are available from this [book's home page](#). This includes a link to the HTML version of this book that mostly uses `object` instead of `img` tags to include images to allow for more detailed alternative text to assist screen readers. The links in this PDF document are colour-coded: internal links are [blue](#), external links are [magenta](#).

The topics covered by this book range from fairly basic (assumes you know how to load a document class and packages) to advanced. To help you navigate your way around this book, sections have symbols to denote the difficulty level. If you only want to learn how to do straight-forward tasks, such as writing a letter without looking up data, you can skip the harder sections. The symbols are as follows:

-  Basic concepts. This may include common L^AT_EX commands described in the earlier volumes or fairly basic commands defined by a simple class or package.
-  Intermediate. This may include more complicated class or package commands, or there may be a wide range of settings (typically `key=value lists`) which can appear a little bewildering at first glance. You may also need to use external applications as part of the document build.

- Advanced. This may include core TeX commands, internal L^AT_EX kernel commands, or programming concepts.

Most chapters start with the basic or intermediate symbol, but they may progress to harder sections. Some of the exercises have a “More Adventurous” part, which increases the difficulty level. There is, of course, a certain amount of subjectivity in choosing the classifications for each section. What one person may find straight-forward, may be more difficult to understand for someone else, so these are just general guides.

To refresh your memory or for those who haven’t read other volumes in this series, throughout this book source code is illustrated in a typewriter font with the word `Input` placed in the margin, and the corresponding output (how it will appear in the PDF document) is typeset with the word `Output` in the margin.

EXAMPLE:

A single line of code is displayed like this:

This is an `\textbf{example}`.

`Input`

The corresponding output is illustrated like this:

This is an **example**.

`Output`

Chapter 1. Introduction

Segments of code that are longer than one line are bounded above and below, illustrated as follows:

```
Line one\par  
Line two\par  
Line three.
```

↑ Input

with corresponding output:

```
Line one  
Line two  
Line three.
```

↑ Output

↓ Output

(Commands typeset in blue, such as `\par`, indicate a hyperlink to the command definition in the [summary](#).)

Command definitions are shown in a typewriter font in the form:

\documentclass[*options*]{*class file*}

Definition

In this case the command being defined is called \documentclass and text typed *like this* (such as *options* and *class file*) indicates the type of thing you need to substitute. (Don't type the angle brackets!) For example, if you want the scrbook class [47] you would substitute *class file* with scrbook and if you want the letterpaper option you would substitute *options* with letterpaper, like this:

\documentclass[letterpaper]{scrbook}

Input

When it's important to indicate a space, the visible space symbol `_` is used. For example:

A_sentence_consisting_of_six_words.

Input

When you type up the code, replace any occurrences of `_` with a space.

[FAQ: Spaces in macros]

Recall from Volume 1 [93, §2] that the comment character `%` is often used to suppress unwanted space caused by the end of line (EOL) character in the source code. For example:

```
Foo%
```

↑ Input

```
Bar
```

↓ Input

1.1 Packages and Document Classes

produces:

FooBar

Output

Any applications that need to be run from a command prompt or terminal (see [Volume 1 \[93, §2.5\]](#)) are displayed in the form:

```
pdflatex mydocument.tex
```

Shell

These should be typed at the command prompt not in your L^AT_EX document.

If a line of code must be typed without any EOL characters but is too long to display on the page, the symbol ↵ is used to indicate a line wrap. Make sure you don't insert a line break at that point.

1.1 Packages and Document Classes

If you already have some basic knowledge of L^AT_EX, you'll know that a class is loaded using

```
\documentclass[<options>]{<name>}
```

Input

and packages are loaded using

1.1 Packages and Document Classes

\usepackage[*options*]{*name*}

Input

If you are using an operating system with case-sensitive filenames (for example, Unix-like systems), the *name* part must use the same case as the corresponding filename *name*.cls or *name*.sty. If the case doesn't match, the file won't be found. In the instance of operating systems with case-insensitive filenames (such as Windows) the file will be found but there will most likely be a warning that *name* doesn't match the declared class or package name. To assist you, this book will always match the filename case when referring to a class or package, rather than the format used in the class or package's manual. (For example, pgf/tikz rather than PGF/TikZ or pstricks rather than PSTricks.) If I use any upper case characters (for example, Alegreya or DejaVuSerif) then that means the filename actually contains those upper case characters.

Recall from [Volume 1](#) [93, §1.1] that package and class documentation can be accessed via the `texdoc` application. For example:

```
texdoc datatool
```

Shell

In the past couple of years that I've been writing this book, I've submitted more bug reports than I have over the previous twenty or more years of using \LaTeX . If it's of any consolation to my fellow coders, I found more bugs in my own code than anyone else's. However this leads me onto a

1.1 Packages and Document Classes

topic I haven't really covered in the previous two volumes (except, perhaps, briefly in [Volume 1 \[93, §C\]](#)) and that's the difference between commercial software and open source software (aside from the obvious one of price).

When a company decides to produce a new piece of software, they usually hire someone to do some market research to find out what potential customers might need, then they hire a programmer (or group of programmers) to write the code and then the software gets tested by a group of product testers. After the buyer has purchased the software, they usually have access to a support desk, customer helpline or chat. The person on the support desk may just have a crib sheet of stock solutions for common issues and if the issue isn't on that list, the customer will probably be referred to a more technical support person. In a few years' time, the company may decide it's no longer financially viable to continue to provide or support that product. If the customer upgrades their operating system (or buys a new computer with a newer operating system) the software may no longer be able to run (or even install), and customers may find themselves in a position where they can no longer access their data that's stored in a proprietary format.

Open source software, on the other hand, usually starts life when someone has a need for an application or piece of code that doesn't exist or exists but is beyond their budget or exists but doesn't run on their operating system. If that person happens to be able to write code (or, perhaps, decides

1.1 Packages and Document Classes

to take this opportunity to learn) then they may be able to write something that fixes their problem. If it achieves what they wanted to do, there's a chance it might be useful to someone else with a similar requirement, so they make it publicly available with an open source licence and an "as is" caveat. There are two most likely possibilities: firstly, once the task has been completed, the developer loses interest in the code they wrote or, secondly, the developer makes repeated long-term use of the code they wrote.

Consider the first case. There's no support desk or customer helpline for this code. The developer has other tasks to do and possibly a full-time job to keep them busy. There are few employers who will pay employees to work full-time on something that doesn't generate an income or improve productivity. If your problem isn't quite identical to the developer's, that code might not work. It may need a few patches or adjustments. Since it's open source, if you understand the code you may be able to modify it yourself. If you don't understand it, you might be able to find someone else who can. This kind of tweaking and patching isn't possible with commercial software. However, it may be that you can't fix it yourself and you can't find anyone willing to help you. At which point you may have to look around for another alternative, which can be time-consuming.

Now consider the second case. Again, there's no support desk or customer helpline, but if the developer is regularly using this code in their

1.1 Packages and Document Classes

work then they have more interest in maintaining and enhancing it. While the code doesn't actually generate any income, it may improve productivity in which case an employer may be more amenable to the developer spending time on updating it. In this case you're in a better position if you use the code and find it doesn't quite work for you.

Although I said that in both cases there's no support desk or customer helpline, there's generally something better: a collection of enthusiastic users who are keen to help others in their spare time. These aren't people just reading stock answers from a crib sheet, they are people who regularly use the code and understand it (the user commands, at least, if not the underlying internals). In the \TeX world, you can find them at places like [\TeX on StackExchange](#), [The L\AT\EX Community](#) or similar forums or on a newsgroup such as `comp.text.tex` [16] or a mailing list (such as the [\TeX User Group \(TUG\)](#) lists at <http://tug.org/mailman/listinfo>). These people are usually friendly, as long as you remember that they are volunteers not employees and that repeating the same old answers is tiresome.

Some of the topics covered in this book suggest a number of alternative classes or packages that you could use. When presented with a selection like this, how do you decide which of them fall under the first case and which the second case? The second case is clearly preferable as any issues are more likely to be fixed. You don't really want to invest time on learning how to use a package only to discover there are ancient unresolved bugs

1.1 Packages and Document Classes

in it.

The first thing to do is check the version number and release date. This information is listed on [the Comprehensive TeX Archive Network \(CTAN\)](#). For example, if you go to <http://ctan.org/pkg/datatool> you'll see the version number listed in the summary. The release date is in both the README file and at the start of the user guide. If the version number is low, such as 1.0, and the release date is some years ago then the package hasn't had much activity. This may not necessarily be a sign that the package falls under the first case as it may be small enough for bugs to be easily detected with a simple test file, but it may well be an indication that the author created it for a particular document, made it publicly available, but hasn't required it for any new documents. The other thing to look for is a "changes" or "history" file. A long revision list is usually an indication of the second case. (Don't mistake a long list of bug fixes as a bad thing. A package that has ten boolean options requires $2^{10} = 1024$ test files to check that all combinations work correctly, and that doesn't include the possibility of conflicts with other packages.)

The next thing to do is a quick search for the package on one of the sites mentioned above. For example, enter the package name in the search box for [TeX on StackExchange](#) or the [ETEX Community Forum](#). Are there many results? If there aren't, that could mean there's not much interest in the package. (Unless, again, it's a very small package that's too trivial

1.2 Arara

to cause any issues.) If, on the other hand, the result list is long and most of the answers involve hacks using [internal commands](#), then the package most likely falls under the first case. If the result list is long and most of the answers don't use internal commands or indicate that there was a bug that has now been fixed or a new feature has been added that solves the problem, then the package most likely falls under the second case. A little time spent on this type of investigation could save you a lot of grief later on.

Occasionally a package or class that has been under the second case for some years suddenly loses maintenance. This is most likely due to a change of circumstances for the developer. (None of us are immortal or immune to adversity or the ageing process!) Fortunately, in the case of some popular classes or packages other volunteers have agreed to take over maintenance. The open source nature of the code makes this more likely than it would with proprietary code.

1.2 Arara

[Volume 2](#) [96, §1.1.2] introduced [arara](#), a Java application that automates the process of building a \LaTeX document. Since then, [arara](#) has evolved into version 4.0 that has useful new features, including conditionals that can be

1.2 Arara

used to determine whether or not an application needs to be run.

To refresh your memory, or for those who haven't read [Volume 2](#) [96, §1.1.2], you need to tell `arara` how to build your document by placing directives as comments within your source code.

EXAMPLE:

```
% arara: pdflatex
\documentclass{article}

\begin{document}
\section{A Sample Section}
This is a sample document.
\end{document}
```

↑ Input

↓ Input

The first line of the source code has a directive:

```
% arara: pdflatex
```

Input

This indicates that `arara` needs to run `pdflatex` so if the document file is called, say, `myDoc.tex`, then the PDF file can be built using:

1.2 Arara

```
arara myDoc
```

Shell

However, it may be that you only want to run pdflatex if the source code has changed. With the new `arara` version 4.0, you can now add a conditional to the directive:

```
% arara: pdflatex if changed("tex")
```

Input

Now `arara` will only run pdflatex if the `.tex` file has changed. You can combine tests using `||` (boolean or) `&&` (boolean and) or `!` (negation). For example:

```
% arara: pdflatex if changed("tex") || missing("pdf")
```

Input

This will now run pdflatex if the `.tex` file has changed or if the PDF file is missing.

In addition to `if` (and the logically opposite `unless`) you can also have loops using `while` (repeat while the condition is true) or `until` (repeat until the condition is true).

Available tests that can be used in the conditions are listed below. The argument `<file ref>` indicates either an extension, such as "log", or a file reference, such as `toFile("xmpl.bib")`. In the case of just a file extension, such as "log", the base is the same as the base of the source `.tex`

1.2 Arara

file that contains the `arara` directives. The argument `<regex>` indicates a regular expression.¹

<code>changed(<file ref>)</code>	Returns true if the file has changed.
<code>unchanged(<file ref>)</code>	Returns true if the file hasn't changed.
<code>exists(<file ref>)</code>	Returns true if the file exists.
<code>missing(<file ref>)</code>	Returns true if the file doesn't exist.
<code>found(<file ref>, <regex>)</code>	Returns true if the regular expression is found in the file.

EXAMPLE:

With earlier versions of `arara`, if your document had cross-references you needed two `pdflatex` directives:

¹See [the Java Pattern class documentation](#) for further details.

1.2 Arara

↑ Input

```
% arara: pdflatex
% arara: pdflatex
\documentclass{article}

\begin{document}
\section{A Sample Section}
Here is a cross-reference to
section~\ref{sec:another}.
```

```
\section{Another Section}
\label{sec:another}
\end{document}
```

↓ Input

However, if you make a change that doesn't affect the cross-references, you only need one invocation of pdflatex. The log file contains information if a rerun is needed:

LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed. Rerun to get
cross-references right.

1.2 Arara

So you could check for “undefined” like this:

```
% arara: pdflatex if changed("tex") || missing("pdf")
% arara: pdflatex if found("log", "undefined")
```

↑ Input

↓ Input

A more general purpose expression to search for is “Rerun”, which may also be used by some packages if they detect a rerun is required that is unrelated to cross-references.

These tests can be combined in a loop. This can make for a long line in your source code, but luckily another new feature of **arara** version 4.0 is the ability to break the line using:

```
% arara: -->
```

Input

at the start of each continuation line. For example:

```
% arara: pdflatex while (changed("tex"))
% arara: --> || missing("pdf")
% arara: --> || missing("log")
% arara: --> || found("log", "Rerun"))
```

↑ Input

↓ Input

1.2 Arara

Now `arara` will repeatedly run `pdflatex` while the `.pdf` or `.log` files are missing or while the `.tex` file has changed or while the log file contains "Rerun". To prevent an infinite process occurring, `arara` will break the loop if the condition is still true after ten iterations. (This maximum value can be changed, if required.)

The above example provides a single `pdflatex` directive in the source code, which is useful if no other applications are required, but this may need to be broken up into multiple directives if another application, such as `bibtex`, needs to be run.

EXAMPLE:

This example has a citation from the sample `xampl.bib` database. (This sample bibliography file should be provided with all modern TeX distributions.)

↑ Input

```
% arara: pdflatex if changed("tex") || missing("pdf")
% arara: bibtex if (missing("bbl") || found("log", "Citation"))
% arara: pdflatex while (found("log", "Rerun"))
\documentclass{article}

\begin{document}
```

1.2 Arara

```
\section{A Sample Section}
Here is a cross-reference to
\ref{sec:another}.
A citation~\cite{book-minimal}.
```

```
\section{Another Section}
\label{sec:another}
```

```
\bibliographystyle{plain}
\bibliography{xmpl}
\end{document}
```

↓ Input

Now `arara` will only run `bibtex` if the `.bbl` file is missing or if the log file contains “Citation”, which will pick up the undefined citation warning:

LaTeX Warning: Citation ‘book-minimal’ on page 1 undefined

Another possible test would be for any changes to the `.bib` file. In this case, the `.bib` file doesn’t have the same root as the main `.tex` file, so I can’t just do `changed("bib")`. Instead I need to indicate that I’m specifying the actual file rather than the extension. This can be done using

1.2 Arara

`changed(toFile("<filename>"))`

Input

where `<filename>` is the name of the file.

In this case, I'm using the `xampl.bib` file, which is in my TeX Live distribution so I'd need the full path name, but if it was in the same directory as my `.tex` file I would just need to do:

`changed(toFile("xampl.bib"))`

Input

 The boolean operators `||` and `&&` are short-circuited. This means that under certain circumstances, only the left-hand condition may be evaluated. For example, if you have `<test1> || <test2>` then if `<test1>` is true, there's no need to evaluate `<test2>` (since both the boolean operations ("true" or "false") and ("true" or "true") evaluate to "true"). Similarly, if you have `<test1> && <test2>` then if `<test1>` is false, there's no need to evaluate `<test2>` (since both ("false" and "false") and ("false" and "true") evaluate to "false"). You may therefore need to consider the optimal ordering of any conditionals, such as placing `changed` or `unchanged` at the start of the condition. Alternatively, you can use the non-short-circuited operators `|` or `&`.

2. ■ MANAGING DATA

Many of the topics covered in this book have examples that fetch row-and-column style data from an external source, such as in a spreadsheet or [structured query language \(SQL\)](#) database. For example, in [§3 Correspondence](#) you may want to pull the recipient's details from a database or you may want to send a template letter to everyone listed in a spreadsheet (mail merging).

This chapter covers accessing data from a [CSV](#) file or spreadsheet or from a MySQL database (the most popular open source database [\[61\]](#)). The MySQL Community Edition is freely downloadable, but there are also commercial versions available. The free version can be fetched from <http://dev.mysql.com/downloads/mysql/>. In addition, this chapter also includes sections on topics that aren't specific to databases, but describe useful utility commands that are used in some of the examples and exercises throughout this book.

 Some of the topics covered in this chapter are quite advanced. If you don't need to fetch data from an external source—for example if you want to write a letter to someone but intend to explicitly write the recipient's

address in your `.tex` file — then you can omit this chapter and skip ahead to §3 Correspondence.

There are a number of packages and applications available on CTAN that can be used to fetch data, see the “Databases” of the TeX Catalogue Topic Index [25]. This book covers the `datatool` bundle (available on modern TeX distributions such as TeX Live and MiKTeX¹) and the `datatooltk` Java application (the installer and source are available on CTAN). This book assumes you have the latest version of `datatool` installed (version 2.23 at time of writing). If you want to try any of the `datatooltk` examples here, ensure you are using the latest version (currently 1.6).

 If you are used to writing programming languages, take care with TeX. Although TeX is Turing-complete, it is very different to most programming languages. (In fact, it's a document formatting language rather than a programming language.) In, say, C or Java the source code is a series of assignments or function calls or commands. Data is explicitly assigned to variables. In TeX the source code consists of data interspersed with control sequences.² Whitespace forms part of the data (except at the start of lines or after control words). In most programming languages `foo(bar)` is

¹TeX Live is included in the MacTeX distribution, and MiKTeX is included in the ProTeXt distribution.

²Although LATEX adds the restriction that document text is only permitted within the document environment.

2.1 Utility Commands

equivalent to `foo(bar)` but in TeX (and L^AT_EX) `\foo{ bar }` is usually not the same as `\foo{bar}`.³ This is because the entire contents of the argument is data not a variable or list of variables. You may not like or agree with TeX's unusual syntax, but no one has so far produced a viable alternative to TeX that has the typesetting power and flexibility of TeX but the structured coding of popular programming languages (although LuaTeX comes close). *Unless told otherwise, assume that spaces aren't ignored.*

2.1 ■ Utility Commands

This section describes utility commands that aren't specific to databases, but are useful when dealing with automation. §2.1.1 describes ways of defining new commands, §2.1.2 describes how to append or prepend code to existing commands and §2.1.3 describes commands that allow you to perform arithmetic calculations. There are some other utility commands concerned with iterating through lists, but these are described later in §2.7.

³Try doing `\begin{ document }` instead of `\begin{document}` in a L^AT_EX document.

2.1 Utility Commands

2.1.1 Macro Definitions

Volume 1 [93, §8] described how to define new commands using `\newcommand` and how to redefine them using `\renewcommand`. However there are times when these commands are too limited for the task in hand. For example, these commands only have a local effect, which means that if you use them to define a command within a group (see Volume 1 [93, §2.7]) they cease to be defined when the group ends. There are also cases where you might want to define a command without checking if it has already been defined. For example, suppose you want to define a “scratch” command that’s used for temporary storage. If you use `\renewcommand`, then you must first initialise the scratch command with `\newcommand`, but if it turns out you never need to use the scratch command, then this definition is redundant and a waste of resources. In this case it’s better to define the scratch command only when it’s needed without the existence checks imposed by `\newcommand` and `\renewcommand`.

[FAQ: Optional arguments like `\section`]

 Since many of the macros described in this section don’t check for the prior existence of the command being defined, it’s your responsibility to ensure you don’t accidentally overwrite a core command which could mess up your document. Some class and package writers use a prefixing system for their command names to help prevent this.

If you’re worried about accidentally overwriting a core command, you

2.1 Utility Commands

can get T_EX to tell you the definition of a command using:

\show <token>

Definition

This will interrupt the document build (as though an error had been encountered) and display the definition of <token> in the transcript. This command can be used for testing purposes but it should be removed once it's provided you with the information you require. The token may be a control sequence or a character, including special characters.

EXAMPLES

1. Suppose you're thinking about defining a command called \c but you first want to find out if it already exists:

```
\documentclass{article}
```

↑ Input

```
\show\c
```

```
\begin{document}
```

```
\end{document}
```

↓ Input

2.1 Utility Commands

When you run L^AT_EX on this document, the transcript will show:

```
> \c=macro:  
->\OT1-cmd \c \OT1\c .  
1.3 \show\c  
  
?
```

This tells you that `\c` is a macro (first line of the above) and it then gives you the definition (second line of the above between `->` and the terminating full stop). The rest of the message is just the regular error message prompt, including the line number. If you type `h` at the prompt for help, it will tell you that this isn't actually an error message.

2. Suppose you're now thinking about defining a command called `\kern` but you first want to find out if it already exists:

```
\documentclass{article}
```

↑ Input

2.1 Utility Commands

```
\show\kern  
  
\begin{document}  
  
\end{document}
```

↓ Input

The transcript now shows:

```
> \kern=\kern.  
1.3 \show\kern
```

?

This may seem a bit strange as it's not showing you a definition, but in this case (La)TeX is telling you that `\kern` is a **primitive** (a core command that's not defined in terms of any other command). If you try redefining this particular command, the results will be disastrous.

3. Suppose you're now thinking about defining a command called `\tmp` but you first want to find out if it already exists:

2.1 Utility Commands

```
\documentclass{article}

\show\tmp

\begin{document}

\end{document}
```

↑ Input

↓ Input

The transcript now shows:

```
> \tmp=undefined.
1.3 \show\tmp

?
```

This means that `\tmp` is undefined.

There's a convenient Perl script called `texdef` that will display the definition of a command without the need for you to use `\show` in your document.
Syntax:

2.1 Utility Commands

```
texdef <options> <cs name>
```

Shell

where *<cs name>* is the name of the command you want to check without the leading backslash. There are a number of options you can use. For example:

- **-t <format>**

This indicates the TeX format. For example, **-t latex** means you're checking a L^AT_EX command. On some systems, there's a convenient shortcut script:

```
latexdef <options> <cs name>
```

Shell

which is equivalent to:

```
texdef -t latex <options> <cs name>
```

Shell

- **-c <cls name>**

This indicates that you want to check if the command is defined in a document that uses the class *<cls name>*.

2.1 Utility Commands

- `-p <sty name>`

This indicates that you want to check if the command is defined in a document that uses the package `<sty name>`.

For a complete list of options either use:

```
texdef --help
```

Shell

or

```
texdoc texdef
```

Shell

EXAMPLE

Suppose you want to know the definition of the command `\forlistloop` provided by the `etoolbox` package:

```
texdef -t latex -p etoolbox forlistloop
```

Shell

This displays:

`\forlistloop:`

`macro:#1#2->\expandafter \etb@forlistloop \expandafter {#2}{#1}`

2.1 Utility Commands

This means that `\forlistloop` is a macro that has two arguments (indicated by `macro:#1#2`) and the code after the arrow `->` indicates the definition of `\forlistloop`.

What about environments? The start of an environment `<envname>`, issued by the command `\begin{<envname>}`, performs some checks, starts a group, stores the environment name in `\@currenvir` and does `\(<envname>)`. The end of the environment `\end{<envname>}` checks that `<envname>` matches `\@currenvir`, does `\end{<envname>}` if that command exists, and then closes the group. (This, incidentally, is why `\newcommand` won't allow you to define a command where the name starts with `end` as it may interfere with the environment end mechanism.)

EXAMPLE

Suppose you want to know the definition of the `figure` environment:

```
texdef -t latex figure endfigure
```

Shell

This displays:

```
\figure:  
\long macro:->\@float {figure}
```

2.1 Utility Commands

```
\endfigure:  
\long macro:->\end@float
```

Alternatively you can use the -E switch:

```
texdef -t latex -E figure
```

Shell

which produces the same output.

Recall from [Volume 1](#) [93, §4.5.1] that the font declarations can also be used as environments. In this case, `\end<envname>` doesn't exist so `\end{<envname>}` skips it.

EXAMPLE

Suppose you want to know the definition of the `Large` environment:

```
texdef -t latex Large endLarge
```

Shell

This displays:

```
\Large:  
\long macro:->@\setfontsize \Large \@xivpt {18}
```

```
\endLarge:  
undefined
```

2.1 Utility Commands

This shows the definition of `\Large` but shows that `\endLarge` is undefined, which means `\Large` can also be used as a declaration.

Now that you know how to check that your chosen macro name doesn't already exist, so you won't accidentally cause a perplexing catastrophic error in your document, the rest of this section will look at defining commands, while the next section ([§2.1.2](#)) will look at modifying existing commands.

`\let<new token><token>`

Definition

This command makes `<new token>` have the same definition as `<token>`.

EXAMPLE:

```
\let\myemph\emph  
\myemph{Test}
```

↑ Input

↓ Input

produces:

Test

Output

Note that this isn't the same as defining `\myemph` to use `\emph`. If `\emph` is later redefined, `\myemph` isn't affected.

2.1 Utility Commands

EXAMPLE:

```
\let\myemph\emph  
\myemph{Test}.
```

↑ Input

```
\renewcommand{\emph}[1]{\textbf{#1}}  
\myemph{Test}. \emph{Test}.
```

↓ Input

produces:

```
Test.  
Test. Test.
```

↑ Output

The command `\let` may be prefixed with `\global` to make the assignment have a global effect.

2.1 Utility Commands

EXAMPLE:

```
\newcommand{\myemph}[1]{\texttt{#1}}
\myemph{Test}. \emph{Test}.
```

↑ Input

```
{% start a group
\global\let\myemph\emph
\myemph{Test}.
```

```
\renewcommand{\emph}[1]{\textbf{#1}}%
\myemph{Test}. \emph{Test}.
```

```
}% end the group
```

```
\myemph{Test}. \emph{Test}.
```

↓ Input

produces:

```
Test. Test.
Test.
```

↑ Output

2.1 Utility Commands

Test. Test.

Test. Test.

↓ Output

In this case, `\myemph` has retained its new definition after the end of the group, but the original definition of `\emph` has been restored, because the effect of `\renewcommand` only lasted until the end of the group.

`\providetcommand{<cs>}[<n-args>][<default>]{<definition>}`

Definition

This is like `\newcommand` except that it will only define the command *if it doesn't already exist*. If `<cs>` already exists, no change will be made to it. The syntax is the same as for `\newcommand`.

EXAMPLE:

↑ Input

```
\providetcommand{\emph}[1]{\textbf{\#1}}%
\emph{Test}.
```

↓ Input

This produces:

2.1 Utility Commands

Test.

Output

Since `\emph` already exists, `\providecommand` had no effect. Compare this to:

```
\providecommand{\Emph}[1]{\textbf{#1}}%
\Emph{Test}.
```

↑ Input

↓ Input

which produces:

Test.

Output

In this case `\Emph` didn't exist, so it was okay for `\providecommand` to define it.

```
\def<cs><arg syntax>{<definition>}
```

Definition

This defines `<cs>` without checking if it already exists. The `<definition>` part is the same as that for `\newcommand`, but with `\def` you don't just specify the number of arguments. Instead you declare the argument syntax in `<arg syntax>` where each parameter is identified by `#<n>` (where `<n>` is a number from 1 to 9).

2.1 Utility Commands

EXAMPLES:

1. The simplest form is when you define a command that has no arguments. For example:

```
\def\test{This is a test.}% definition  
\test
```

↑ Input

↓ Input

produces:

This is a test.

Output

2. This example defines a command with two arguments:

```
\def\test#1#2{\emph{#1} \textbf{#2}}%  
\test{First}{Second}
```

↑ Input

↓ Input

2.1 Utility Commands

This produces:

First Second

Output

3. This example defines a rather more complicated command that has the syntax:

```
\test <arg1>-<arg2>-<arg3>\endtest
```

This parameter syntax is now #1-#2-#3\endtest which is in the *<arg syntax>* part below:

```
\def\test#1-#2-#3\endtest{\emph{#1} \textbf{#2} \texttt{#3}}
```

Input

Note that this doesn't define a command called \endtest. The \endtest token merely forms part of the argument syntax. This \test command can now be used but only with the correct syntax:

```
\test First-Second-Third\endtest
```

Input

This produces:

2.1 Utility Commands

First Second Third

Output

If you attempt to use incorrect syntax, for example, if you try

\test{First}{Second}{Third}



then you will get a runaway argument error:

```
Runaway argument?  
{First}{Second}{Third}  
! Paragraph ended before \test was complete.
```

 Within $\langle arg\ syntax \rangle$ the parameters must appear in ascending order (for example, #2#1 is invalid); parameters can't be repeated (for example, #1#1 is invalid); intermediate parameters can't be skipped out (for example, #1#3 is invalid). This is a restriction on $\langle arg\ syntax \rangle$ not on $\langle definition \rangle$.

\def only has a local effect, like \newcommand. If you want to globally define a command, then you can use the analogous:

\gdef<cs><arg syntax>{<definition>}

Definition

There are two more similar commands:

2.1 Utility Commands

\edef<cs><arg syntax>{<definition>}

Definition

and

\xdef<cs><arg syntax>{<definition>}

Definition

In both cases, the command being defined is set to the full expansion of <definition>. The first, \edef, only has a local effect. The second, \xdef, has a global effect.

EXAMPLE:

This example illustrates the difference between the non-expansion \gdef and the expansion \xdef:

```
\def\abc{abc}
\def\xyz{xyz}
% start group
\def\abc{ABC}
\def\xyz{XYZ}
\gdef\test{\abc; \xyz}
\xdef\etest{\abc; \xyz}
\test. \etest.
```

↑ Input

2.1 Utility Commands

}%

\test. \etest.

↓ Input

This produces:

ABC; XYZ. ABC; XYZ.
abc; xyz. ABC; XYZ.

↑ Output

↓ Output

In this example, `\test` was just defined in terms of `\abc` and `\xyz`, so when `\test` is executed, it uses the current definitions of those commands. On the other hand, `\etest` was defined as the full expansion of `\abc` and `\xyz`. This means that if the definitions of `\abc` and `\xyz` later change, `\etest` is unaffected.

⚠ Take care when using `\edef` and `\xdef` that the definition doesn't contain fragile commands, as that can cause an error.

If you want to protect against fragile commands, L^AT_EX provides additional commands analogous to `\edef` and `\xdef`:

2.1 Utility Commands

`\protected@edef{cs}{arg syntax}{definition}`

Definition

and

`\protected@xdef{cs}{arg syntax}{definition}`

Definition

Note that these are **internal commands**, so if you need to use them in your document (rather than in a class or package), you need `\makeatletter` and `\makeatother`.

[FAQ: `\@` and `@` in macro names]

Note that `\def`, `\gdef`, `\edef` and `\xdef` all define short commands, unless the prefix `\long` is used. For example:

`\long\def\test#1{\emph{#1}}`

Input

This now allows a paragraph break within the argument of `\test`. For further details see *The TeXbook* [46]. (Both `\protected@edef` and `\protected@xdef` also define short commands, but they can't be prefixed with `\long`.)

In addition to the TeX **primitives** and core L^AT_EX commands described above, the `etoolbox` package [51] also provides ways of defining commands.

`\csdef{cs name}{arg syntax}{definition}`

Definition

This is analogous to `\def` except that you supply the control sequence name `{cs name}` (without the leading backslash) instead of the actual control

2.1 Utility Commands

sequence. Note that $\langle cs\ name \rangle$ must be fully expandable. The $\langle arg\ syntax \rangle$ is the same as for [`\def`](#).

Defining a control sequence by its name in this manner means that you can have control sequences that include punctuation or digits or you can form the control sequence name on the fly. This is the way that commands with labels work. For example, recall the [`\newglossaryentry`](#) command from [Volume 2 \[96, §6.1.2\]](#). The first argument is a label, and this label is used in the name of the internal control sequences that store the entry details. The `datatool` package's database commands described later in this chapter also use a similar technique.

The `etoolbox` package also provides:

`\csgdef{\langle cs name \rangle}{\langle arg syntax \rangle}{\langle definition \rangle}`

Definition

analogous to [`\gdef`](#),

`\csedef{\langle cs name \rangle}{\langle arg syntax \rangle}{\langle definition \rangle}`

Definition

analogous to [`\edef`](#),

`\csxdef{\langle cs name \rangle}{\langle arg syntax \rangle}{\langle definition \rangle}`

Definition

analogous to [`\xdef`](#),

2.1 Utility Commands

\protected@csedef{*cs name*}{*arg syntax*}{*definition*}

Definition

analogous to \protected@edef, and

\protected@csxdef{*cs name*}{*arg syntax*}{*definition*}

Definition

analogous to \protected@xdef.

There are also commands analogous to \let:

\cslet{*cs name*}{{*cs*}}

Definition

\letcs{*cs*}{{*cs name*}}

Definition

and

\csletcs{*cs name*}{{*cs name*}}

Definition

where *cs name* is the name of a control sequence and *cs* is a control sequence.

You can use a control sequence by its name with:

\csuse{*cs name*}

Definition

Remember that if the control sequence has been defined to have arguments, you need to specify these afterwards. For example:

2.1 Utility Commands

`\csuse{section}{A Sample Section}`

Input

is equivalent to:

`\section{A Sample Section}`

Input

If the control sequence doesn't exist, `\csuse{(cs name)}` will expand to nothing. An alternative is to use TeX's primitive:

`\csname <cs name>\endcsname`

Definition

If the control sequence doesn't exist, `\csname` will define the control sequence to `\relax` before using it.

The etoolbox package also provides ways of determining if a command already exists using:

`\ifdef{<cs>}{{<true part>}}{{<false part>}}`

Definition

where `<cs>` is the control sequence whose existence is being tested. Alternative you can specify the control sequence name using:

`\ifcsdef{<cs name>}{{<true part>}}{{<false part>}}`

Definition

In both cases, if the command has been defined, `<true part>` will be processed, otherwise `<false part>` will be processed. You can also use the logically opposite commands which check for non-existence:

2.1 Utility Commands

`\ifundefined{\cs}{\{true part\}}{\{false part\}}`

Definition

where $\langle cs \rangle$ is a control sequence and

`\ifcsundef{\cs name}{\{true part\}}{\{false part\}}`

Definition

where $\langle cs \, name \rangle$ is a control sequence name. Although logically opposite, these commands differ slightly from their existence counterparts as these commands will consider a control sequence undefined if its definition is `\relax`.

If you don't want to use the `etoolbox` package, the L^AT_EX kernel provides the [internal commands](#):

`\@ifundefined{\cs name}{\{true part\}}{\{false part\}}`

Definition

which is similar to `\ifcsundef`,

`\@nameuse{\cs name}`

Definition

which is equivalent to

`\csname \cs name\endcsname`

Input

and

2.1 Utility Commands

\@namedef{\⟨cs name⟩}{⟨arg syntax⟩}{⟨definition⟩}

Definition

which is similar to \csdef.

2.1.2 ■ Hook Management

The etoolbox package [51] comes with some useful hook management tools that allow you to append (or prepend) code to a control sequence's definition. To append ⟨code⟩ to the definition of the command ⟨cs⟩:

\appto{\⟨cs⟩}{⟨code⟩}

Definition

or to make the effect global:

\gappto{\⟨cs⟩}{⟨code⟩}

Definition

EXAMPLE

```
\newcommand{\mymacro}{x}%
\mymacro;
\appto{\mymacro}{AB}%
\mymacro;
```

↑ Input

2.1 Utility Commands

```
\appto\mymacro{YZ}%
\mymacro.
```

↓ Input

This produces:

x; xAB; xABYZ.

Output

Note that no expansion is performed on `(code)`. For example:

```
\newcommand{\mymacro}{x}%
\newcommand{\myothermacro}{AB}%
\appto\mymacro{\myothermacro}
\renewcommand{\myothermacro}{YZ}%
\appto\mymacro{\myothermacro}
```

↑ Input

↓ Input

is equivalent to:

```
\newcommand{\mymacro}{x\myothermacro\myothermacro}
```

Input

When you actually use `\mymacro` it will use whatever the definition of `\myothermacro` is at the time, which may not be what you intended.

2.1 Utility Commands

If you want `<code>` expanded before being appended to `<cs>` then you need to use:

`\eappto{cs}{<code>}`

Definition

or its global equivalent:

`\xappto{cs}{<code>}`

Definition

So

```
\newcommand{\mymacro}{x}%
\newcommand{\myothermacro}{AB}%
\eadpto{\mymacro}{\myothermacro}%
\renewcommand{\myothermacro}{YZ}%
\eadpto{\mymacro}{\myothermacro}
```

↑ Input

↓ Input

is equivalent to:

`\newcommand{\mymacro}{xABYZ}`

Input

When you use any of the versions that expand the item, if the item contains a command that shouldn't be expanded, you need to use:

2.1 Utility Commands

\noexpand⟨cs⟩

Definition

where ⟨cs⟩ is the command that shouldn't be expanded. Fragile commands will typically need to have their expansion suppressed. For example:

```
\newcommand{\mymacro}{x}%
\newcommand{\myothermacro}{AB}%
\appto{\mymacro}{\noexpand\footnote{\myothermacro}}
```

↑ Input

↓ Input

is equivalent to:

```
\newcommand{\mymacro}{x\footnote{AB}}
```

Input

Now the fragile `\footnote` command won't be processed until `\mymacro` is used later on in the document.

There are similar commands for prepending code to a command:

\preto⟨cs⟩{⟨code⟩}

Definition

or its global equivalent:

2.1 Utility Commands

\gpreto{cs}{<code>}

Definition

As with \appto and \gappto, no expansion is performed on <code>. If you want <code> expanded you need to use:

\epreto{cs}{<code>}

Definition

or its global equivalent:

\xpreto{cs}{<code>}

Definition

The expansion commands, such as \eappto and \epreto, *fully expand* <code>. This means that a recursive expansion is performed until everything is expanded except commands that have been prefixed with \noexpand or commands that don't expand, such as primitives or robust commands. For example:

```
\newcommand{\mymacro}{x}%
\newcommand{\mymacroB}{\mymacroC}%
\newcommand{\mymacroC}{\mymacroD}%
\newcommand{\mymacroD}{Z}%
\eappto{\mymacro}{\mymacroB}
```

↑ Input

↓ Input

2.1 Utility Commands

is equivalent to:

```
\newcommand{\mymacro}{x}
```

Input

It may be that you only want one-level of expansion. In which case you can use the etoolbox command:

```
\expandonce\cs
```

Definition

For example:

```
\newcommand{\mymacro}{x}%
\newcommand{\mymacroB}{\mymacroC}%
\newcommand{\mymacroC}{\mymacroD}%
\newcommand{\mymacroD}{Z}%
\appto{\mymacro}{\expandonce{\mymacroB}}
```

↑ Input

↓ Input

is equivalent to:

```
\newcommand{\mymacro}{x\mymacroC}
```

Input

There are also analogous commands that require the name (without the leading backslash) of the control sequence:

2.1 Utility Commands

<code>\csappto{\cs name}{\code}</code>	Definition
(appending without expansion)	
<code>\csgappto{\cs name}{\code}</code>	Definition
(global version)	
<code>\cseappto{\cs name}{\code}</code>	Definition
(appending with expansion)	
<code>\csxappto{\cs name}{\code}</code>	Definition
(global version)	
<code>\cspreto{\cs name}{\code}</code>	Definition
(prepend without expansion)	
<code>\csgpreto{\cs name}{\code}</code>	Definition
(global version)	
<code>\csepreto{\cs name}{\code}</code>	Definition
(prepend with expansion)	

2.1 Utility Commands

`\csxpreto{\(cs name\)}{\(code\)}`

Definition

(global version).

2.1.3 ■ Arithmetic

\TeX count registers only allow integer values. Even the registers that store dimensions (such as `\parindent`) use integer arithmetic as \TeX internally stores lengths in terms of its sp unit ($65\,536\,\text{sp} = 1\,\text{pt}$). There are, however, some packages that allow you to perform decimal calculations using \TeX (see, for example, the [calculation topic](#) or the [arithmetic topic](#)).

However, first let's look at just integer arithmetic, as that can be performed more efficiently using \TeX primitive than using \LaTeX packages. \TeX count registers are defined using:

`\newcount\register cs`

Definition

This not only allocates a new register, but also assigns a control sequence `\register cs` that can be used to reference the register. For example:

`\newcount\mycount`

Input

allocates a new register that can be referenced using `\mycount`.

2.1 Utility Commands

⚠ Be careful not to confuse T_EX's `\newcount` with L_AT_EX's `\newcounter` command. With the L_AT_EX version, you provide a label as the argument. Internally, `\newcounter` uses `\newcount` to create a register where the control sequence is formed from `\c@<label>` (where `<label>` is the argument of `\newcounter`). However, `\newcounter` performs more than simply assigning a new register. It also provides a command that can be used to format the value of the register (`\the<label>`), allows the value to be cross-referenced using L_AT_EX's `\label/\ref` (along with `\refstepcounter`), and can also automatically reset the value of the register when another counter is incremented. If I only want a scratch variable to calculate integer values, I'll just define a register using `\newcount`, as it helps to reduce the clutter of (possibly already complicated code) if I can refer to the register control sequence directly, without the cumbersome use of `\value` or `internal commands`. (See, for example, the `\julianday` register in [Example 38](#).)

A register can be assigned a value using the syntax:

`<register>=<value>`

Definition

For example:

`\mycount = 25`

Input

sets the value of the `\mycount` register to 25. Note that it's sometimes nec-

2.1 Utility Commands

essary to use `\relax` after the assignment to prevent \TeX from prematurely expanding the following `token`. ϵ - \TeX provides a `primitive` for evaluating expressions:

`\numexpr<integer expression>`

Definition

For example:

`\mycount=\numexpr(25+5)/3`

Input

It's usually a good idea to put `\relax` after `<integer expression>` in case it happens to be followed by something that could be interpreted as part of the expression:

```
\mycount=\numexpr(25+5)/3\relax  
-- this assignment is followed by an en-dash.
```

↑ Input

↓ Input

There is a similar ϵ - \TeX `primitive` for evaluating dimension expressions:

`\dimexpr<dimension expression>`

Definition

You can display the value of the register using:

2.1 Utility Commands

`\the<register>`

Definition

For example:

```
\mycount = 25\relax  
\the\mycount
```

↑ Input

↓ Input

produces:

25

Output

This also works with other types of registers. For example:

`\the\textwidth`

Input

produces:

271.30533pt

Output

There's a similar TeX primitive that works on either a register or a number (either typed explicitly or stored in a macro):

2.1 Utility Commands

\number<num>

Definition

For example:

```
\mycount = 25
\newcommand{\mynum}{40}%
Register: \number\mycount.
Macro: \number\mynum.
Number: \number46.
```

↑ Input

↓ Input

Produces:

Register: 25. Macro: 40. Number: 46.

Output

The contents of a register can be incremented using:

\advance<register> by <value>

Definition

where <value> may be another register, a macro that expands to a value or a plain number. The by keyword is optional. For example:

2.1 Utility Commands

```
\mycount = 12  
\advance\mycount by -3  
Result: \number\mycount.
```

↑ Input

↓ Input

produces:

Result: 9.

Output

The contents of a register can be multiplied by a value using:

\multiply<register> by <value>

Definition

As with `\advance`, the keyword `by` may be omitted. For example:

```
\mycount = 12  
\multiply\mycount by 2  
Result: \number\mycount.
```

↑ Input

↓ Input

2.1 Utility Commands

produces:

Result: 24.

Output

The contents of a register can be divided by a value using:

\divide<register> by <value>

Definition

As before the by keyword may be omitted. Remember that this uses integer arithmetic. For example:

```
\mycount = 25
\divide\mycount by 3
Result: \number\mycount.
```

↑ Input

↓ Input

produces:

Result: 8.

Output

For decimal arithmetic, there are a number of packages available, such as fp [57] and pgfmath [102]. In addition, the L^AT_EX3 experimental bundle [50] also provides decimal arithmetic. However, it uses L^AT_EX3 syntax, which is beyond the scope of this book.

[FAQ: L^AT_EX3 programming]

2.1 Utility Commands

The `datatool` package provides an interface to either `fp` or `pgfmath`. The default is to use `fp` but you can change this using the `math=pgfmath` package option:

```
\usepackage[math=pgfmath]{datatool}
```

Input

The `pgfmath` package is part of the `pgf` bundle, so if you intend loading `pgf` (or `tikz`) in your document, it's more efficient to use the `pgfmath` engine with `datatool` to avoid the overhead of loading an additional package.

The `fp` and `pgfmath` packages use very different syntax to perform the same calculations, but `datatool` provides the same interface commands regardless of the underlying arithmetical package, so you can switch engines without having to change your code, however you may find minor differences in the results, caused by different levels of precision or rounding.

There are two types of arithmetical commands provided by `datatool`: those that operate on raw plain numbers that use a full stop as the decimal point with no group separator, and those that operate on locale dependent numbers or currency. The first type are prefixed by `d1l`. For example:

```
\d1lround{\langle cs \rangle}{\langle number \rangle}{\langle num digits \rangle}
```

Definition

This rounds `\langle number \rangle` to `\langle num digits \rangle` decimal places and stores the result in the control sequence `\langle cs \rangle`.

The second type are prefixed by DTL. For example:

2.1 Utility Commands

`\DTLround{⟨cs⟩}{⟨number⟩}{⟨num digits⟩}`

Definition

which is the locale-dependent alternative to `\dtlround`.

The plain versions (prefixed with dtl) all perform local assignments. If you need a global assignment you can use `\global\let` on the result (see §2.1.1). For example:

```
{% local scope
  \dtlround\mynum{14.39999}{2}% perform rounding
  \global\let\mynum\mynum
}
\mynum
```

↑ Input

↓ Input

The locale versions (prefixed with DTL) come with two alternatives: a local version and a global version. The global versions have the prefix DTLg such as:

`\DTLground{⟨cs⟩}{⟨number⟩}{⟨num digits⟩}`

Definition

which is the global alternative to `\DTLround`.

2.1 Utility Commands

For the locale versions, the decimal character defaults to a full stop and the number group separator defaults to a comma, but these can be changed using:

\DTLsetnumberchars{\<number group char>}{\<decimal char>}

Definition

where *<number group char>* is the number group separator and *<decimal char>* is the decimal character. For example, to switch to commas for the decimal character and a full stop for the number group separator:

\DTLsetnumberchars{.}{,}

Input

Now, any numbers that use this format need to use the control sequences prefixed with DTL instead of dtl:

```
\DTLsetnumberchars{.}{,}
\DTLround\mynum{1.250,2398}{2}\mynum.
```

↑ Input

↓ Input

This produces:

1.250,24.

Output

2.1 Utility Commands

EXAMPLE (PLAIN NUMBERS):

```
\dtlround{\mynum}{14.39999999999999}{2}\mynum;  
\dtlround{\mynum}{2.5}{2}\mynum.
```

↑ Input

↓ Input

produces:

14.40; 250.

Output

EXAMPLE (LOCALE NUMBERS):

```
\DTLround{\mynum}{2,014.399999999999}{2}\mynum;  
\DTLround{\mynum}{1,002.5}{2}\mynum.
```

↑ Input

↓ Input

produces:

2.1 Utility Commands

2,014.40; 1,002.50.

Output

The locale version `\DTLround` first converts the formatted number into a plain number, performs the arithmetical operation, and then converts the result back into a formatted number. Therefore, if your numbers are all plain numbers, it's more efficient to use `\dtlround` instead of `\DTLround`. Similarly for all the commands described below.

Addition can be performed using:

`\dtladd{\langle cs \rangle}{\langle number 1 \rangle}{\langle number 2 \rangle}`

Definition

for the plain version, or

`\DTLadd{\langle cs \rangle}{\langle number 1 \rangle}{\langle number 2 \rangle}`

Definition

for the scoped locale version, or

`\DTLgadd{\langle cs \rangle}{\langle number 1 \rangle}{\langle number 2 \rangle}`

Definition

for the global locale version. In each case the sum of `\langle number 1 \rangle` and `\langle number 2 \rangle` is stored in the control sequence `\langle cs \rangle`.

Subtraction can be performed using:

`\dtlsub{\langle cs \rangle}{\langle number 1 \rangle}{\langle number 2 \rangle}`

Definition

for the plain version, or

2.1 Utility Commands

`\DTLsub{<cs>}{<number 1>}{<number 2>}`

Definition

for the scoped locale version, or

`\DTLgsub{<cs>}{<number 1>}{<number 2>}`

Definition

for the global locale version. In each case $<\text{number } 1>$ minus $<\text{number } 2>$ is stored in the control sequence $<\text{cs}>$.

Multiplication can be performed using:

`\dtlmul{<cs>}{<number 1>}{<number 2>}`

Definition

for the plain version, or

`\DTLmul{<cs>}{<number 1>}{<number 2>}`

Definition

for the scoped locale version, or

`\DTLgmul{<cs>}{<number 1>}{<number 2>}`

Definition

for the global locale version. In each case the product of $<\text{number } 1>$ and $<\text{number } 2>$ is stored in the control sequence $<\text{cs}>$.

Division can be performed using:

2.1 Utility Commands

`\dtldiv{<cs>}{<number 1>}{<number 2>}`

Definition

for the plain version, or

`\DTLdiv{<cs>}{<number 1>}{<number 2>}`

Definition

for the scoped locale version, or

`\DTLgdiv{<cs>}{<number 1>}{<number 2>}`

Definition

for the global locale version. In each case `<number 1>` divided by `<number 2>` is stored in the control sequence `<cs>`.

The absolute value can be obtained using:

`\dtlabs{<cs>}{<number>}`

Definition

for the plain version, or

`\DTLabs{<cs>}{<number>}`

Definition

for the scoped locale version, or

`\DTLgabs{<cs>}{<number>}`

Definition

for the global locale version. In each case the absolute value of `<number>` is stored in the control sequence `<cs>`.

The negation can be obtained using:

2.2 Loading Data

`\dtlneg{\cs}{\number}`

Definition

for the plain version, or

`\DTLneg{\cs}{\number}`

Definition

for the scoped locale version, or

`\DTLgneg{\cs}{\number}`

Definition

for the global locale version. In each case the negative of $\langle \text{number} \rangle$ is stored in the control sequence $\langle \text{cs} \rangle$.

There are also commands available to perform arithmetic operations on a column of data stored in one of datatool's internal database. However, these use the DTL versions and since TeX isn't designed for data management, it's better to perform these calculations in your spreadsheet or when you pull the data from a [SQL](#) database.

2.2 □ Loading Data

Before you can use data from an external source in your document, you must first load it. This section describes how to load data from a [CSV](#) file or from a datatool (.dbtex) file. When the data is loaded, it's stored in an

2.2 Loading Data

internal datatool database with an associated label. This label is then used to identify the internal database whenever you want to fetch values from it in your document. Throughout the rest of this book, $\langle db\text{-name} \rangle$ will be used to indicate this label. It's best to just use the letters a, \dots, z, A, \dots, Z or the digits $0, \dots, 9$ within $\langle db\text{-name} \rangle$ to avoid accidentally using problematic [special characters](#).

The database is divided up into columns (or fields) and rows where a column can be referenced either by its index (starting from 1) or by its label (or key) and a row can be referenced by its index (starting from 1). Throughout the rest of this book, $\langle row\text{-idx} \rangle$ will be used to indicate the row index, $\langle col\text{-idx} \rangle$ will be used to indicate the column index and $\langle col\text{-label} \rangle$ will be used to indicate the column label. As with $\langle db\text{-name} \rangle$, $\langle col\text{-label} \rangle$ should not contain any special characters. Since the label is often used in a list context, it's also best to avoid commas.

2.2.1 Loading Data From a CSV File

Most spreadsheet applications can export data to a [CSV](#) file. By default datatool assumes that the data in this file is separated by commas where the values are optionally delimited with the double-quote character " but if this isn't the case you need to specify the separator using:

2.2 Loading Data

\DTLsetseparator{\langle character\rangle}

Definition

and the delimiter using:

\DTLsetdelimiter{\langle character\rangle}

Definition

A common alternative is to use the tab character  as a separator but this is awkward to specify in L^AT_EX, so datatool provides

\DTLsettabspace

Definition

which is the same as \DTLsetseparator{\langle character\rangle} where *⟨character⟩* is a tab. Note that this command changes the **category code** of the tab character. If you later want to treat a tab as a regular space, you can reset the **category code** after you have loaded the data using:

\DTLmaketabspace

Definition

 Remember to specify the separator and delimiter characters *before* you load the file.

For example, suppose your data is saved in a CSV file in the form:

Surname;First Name;Title;Registration Number

|Smith, Jr|;John;Mr;12345

Brown;Jane;Miss;12346

Brown;Andy;12347

2.2 Loading Data

Adams; "oe|

then in your document, before you load this file, you need to write:

\DTLsetseparator{;}\DTLsetdelimiter{|}

Input

For convenience, all the examples in this book assume the default comma separator and double-quote delimiter.

 When creating your CSV files be careful of spurious spaces. For example, the following line of data:

Brown , Jane , Miss , 12356



isn't the same as:

Brown,Jane,Miss,12356



If the CSV file contains extended characters, make sure the file was saved with the same encoding as your L^AT_EX document and use the `inputenc` [40] and `fontenc` [59] packages.⁴ (See Volume 1 [93, §4.3.1].) The sample files that accompany this book all use UTF-8 encoding. Make sure you load the `inputenc` package with the `utf8` option before you load any of these CSV files.

⁴Just use `fontspec` [76] for X_EL^AT_EX.

2.2 Loading Data

Once your data is in a CSV file you can load it into a datatool database using:

`\DTLloaddb[⟨options⟩]{⟨db-name⟩}{⟨filename⟩}`

Definition

where the CSV file is called `⟨filename⟩`. The argument `⟨db-name⟩` is the database label, as described [above](#).

The `\DTLloaddb` command assumes the CSV file either doesn't contain any of TeX's special characters (see [Volume 1 \[93, §4.3\]](#)) or, if it does, they form correct L^AT_EX code. If this isn't the case, instead of using `\DTLloaddb`, you can use:

`\DTLloadrawdb[⟨options⟩]{⟨db-name⟩}{⟨filename⟩}`

Definition

This is like `\DTLloaddb` except that it performs a substitution on nine of the ten special characters. (The backslash always retains its special state.) The mappings are listed in [Table 2.1](#).

You can add extra mappings using:

`\DTLrawmap{⟨string⟩}{⟨replacement⟩}`

Definition

For example, to replace the character £ with `\pounds`:

`\DTLrawmap{£}{\pounds}`

Input

(Alternatively use the `inputenc` package.)

2.2 Loading Data

Table 2.1 Special character mappings used by \DTLloadrawdb

Character	Mapping
%	\%
\$	\\$
&	\&
#	\#
-	_
{	\{
}	\}
~	\textasciitilde
^	\textasciicircum

2.2 Loading Data

EXAMPLES

1. Suppose your CSV file looks like:

```
Experiment,Result  
1,$42.08\pm 0.1$  
2,$48.03\pm 0.2$
```

In this file, the \$ character has been used to indicate in-line maths mode (see Volume 1 [93, §9.1]). This should be left as it is when the data is loaded, so use `\DTLloadaddb`.

2. Suppose your CSV file looks like:

```
Title,Price  
"Duck & Goose's Adventures",\$10.00  
"The Return of Duck & Goose",\$11.00
```

Now the characters & and \$ are intended literally and should not be interpreted by TeX, so you need to use `\DTLloadrawdb` to ensure they are converted to the correct commands.

3. Now suppose your CSV file looks like:

2.2 Loading Data

```
Title,Price  
"Duck & Goose's Adventures",£10.00  
"The Return of Duck & Goose",£11.00
```

Again you need to use `\DTLloadaddb` but before you do that you may need to define a new mapping for the £ character using `\DTLrawmap`, as described above. (Or use the `inputenc` package.)

 The commands `\DTLloadaddb` and `\DTLloadrawdb` can't read data where there are EOL characters within a cell. For example, neither command can read a CSV file that looks like:

```
Title,Price  
"Duck and Goose's  
Adventures",10.00  
"The Return of  
Duck and Goose",11.00
```

If you have a CSV file in this form, you can use `datatooltk` to convert the CSV file to a datatool (.dbtex) file, described in the [next section](#).

You may have noticed that both `\DTLloadaddb` and `\DTLloadrawdb` have an optional argument. This is a [key=value list](#). Available keys are as follows:

noheader This is a [boolean key](#). The value can be either `false` (the CSV file has a header row, as in the examples above) or `true` (the

2.2 Loading Data

CSV file doesn't have a header row). The default is `false`. If you want to set this value you may omit `=true`, so `noheader=true` is the same as just `noheader`.

keys Each column must have a unique label assigned to it. This makes it easier to reference but, like the database label, the column label mustn't contain **special characters**. The default action of `\DTLloaddb` and `\DTLloadrawdb` is to use the value given in the header row as a label. This may be inappropriate, so you can set a different set of labels using this option. The value must be a **comma-separated list** of labels in the same order as the columns in the CSV file. For example,

```
\DTLloaddb[keys={title,price}]{books}{booklist.csv}           Input
```

(note the braces).

You should use this option if the header row contains **special characters**. If the CSV file has no header and no label has been specified, a default label is generated in the form

```
\dtldefaultkey <n>                                         Definition
```

2.2 Loading Data

where $\langle n \rangle$ is the column number. By default, `\dtldefaultkey` is just “Column”, but you can change this by redefining the command (see [Volume 1 \[93, §8.2\]](#)). Note that an empty item in the `keys` list indicates an empty label for that column.

- autokeys** This is a [boolean key](#) that was introduced in version 2.22. If true, all the column labels will automatically be assigned the default label `\dtldefaultkey` $\langle n \rangle$ described above. This is useful if you have a lot of columns where the header may contain [special characters](#), and you don’t want to have to list every column in the `keys` list. This means that you need to know the column index if you want to reference the data in it.
- headers** Each column not only has a unique label assigned to it, but also has a header or title. The column headers are used in commands such as `\DTLdisplaydb` described in [§2.6](#). The default column headers are taken from the header row in the CSV file but if they aren’t appropriate or your file doesn’t have a header row, you can use this option to assign headers. As with the `keys` option, described above, the value must be a [comma-separated list](#) of header text in the same order as the columns in the CSV file. For example:

2.2 Loading Data

↑ Input

```
\DTLloaddb[headers={Book Title,Price (\pounds)}]
{books}% database label
{booklist.csv}% filename
```

↓ Input

(note the braces). An empty item in the headers list indicates an empty header for that column. For example:

```
\DTLloaddb[headers={Book Title,}]{books}{booklist.csv}
```

Input

indicates that the second column has a blank header.

omitlines The value must be a non-negative number indicating how many lines to skip at the start of the CSV file. For example, if the CSV file contains two lines of unwanted material at the start, then you need to use `omitlines=2`.

2.2 Loading Data

EXAMPLES

1. Suppose your CSV file called `products.csv` looks like:

This is a list of products in my shop.

Last edited 2014-01-22

Title,Price (£)

"Duck & Goose's Adventures",10.00

"The Return of Duck & Goose",11.00

When you load this data into your document, you need to skip the first three lines as they don't form part of the data. You also need to map the pound symbol (£) and the ampersand (&). Additionally, it's a good idea to provide short unique labels to identify the columns:

↑ Input

```
\DTLrawmap{f}{\pounds}%
\DTLloadrawdb
[%]
omitlines=3,% header row is on line 4
keys={title,price}% column labels
```

2.2 Loading Data

```
]%  
{products}% database label  
{products.csv}% filename
```

↓ Input

2. Suppose your CSV file called products.csv looks like:

```
"Duck and Goose's Adventures",10.00  
"The Return of Duck and Goose",11.00
```

Here there isn't a header row. You could assign labels as in the previous example:

```
\DTLloaddb  
[%  
noheader,% no header row in file  
keys={title,price},% column labels  
headers={Title,Price (\pounds)}% column titles  
]%
```

```
{products}% database label  
{products.csv}% filename
```

↑ Input

2.2 Loading Data

↓ Input

However, if you're not interested in referencing any columns (for example, you just want to display the data in a table, as described in §2.6) you can let datatool assign default labels:

↑ Input

```
\DTLloaddb
[%  
  noheader,% no header row in file
  headers={Title,Price (\pounds)}% column titles
]%
{products}% database label
{products.csv}% filename
```

↓ Input

2.2.2 Loading Data From a .dbtex File

A datatool (.dbtex) file is a L^AT_EX file with definitions and assignments of macros and registers used to represent an internal database. Although

2.2 Loading Data

this is a plain text file, it's very difficult to read and edit. However it's the fastest way of loading data via datatool. The `datatooltk` application can read and write this file, but it can also import data from CSV files, Excel .xls files, Open Document .ods spreadsheets or MySQL databases. (You can't export back to those formats.) It can be run either in batch mode from a command prompt (see [Volume 1 \[93, §2.5\]](#)) or as a graphical user interface.

It's simple to load a .dbtex file into a document that uses the `datatool` package. Just use:

`\input{<filename>}`

Definition

where `<filename>` is the name of the file including the .dbtex extension. (This must come after `\usepackage{datatool}`.) If you can't remember the label you assigned to the data, you can reference it using:

`\dtllastloadeddb`

Definition

However, a more convenient approach is to use:

`\DTLloaddbtex{<cs>}{<filename>}`

Definition

This inputs `<filename>` and makes the control sequence `<cs>` have the same value as `\dtllastloadeddb`. For example

2.2 Loading Data

```
\DTLloaddbtx{\people}{people.dbtex}
```

Input

works like:

```
\input{people.dbtex}  
\let\people\dtllastloadeddb
```

↑ Input

↓ Input

but it checks for the existence of the file `people.dbtex` and checks that the new command (`\people`) isn't already defined. Using `\DTLloaddbtx` is therefore safer than just using `\input` and `\let`. When you use commands like `\DTLdisplaydb` (§2.6) and `\DTLforeach` (§2.7.1) you can now reference the data using your new command (`\people` in this example).

The column title can be changed using:

```
\DTLsetheader{\<db-name>}{\<col-label>}{\<title>}
```

Definition

where `\<db-name>` is the label identifying the data (`\people` in the above example), `\<col-label>` is the label identifying the required column and `\<title>` is the new column title.

2.2 Loading Data

EXAMPLE 4. CONVERT A .CSV FILE TO A .DBTEX FILE

 The default CSV escape character is a backslash, which means that if your data contains any control sequences the backslash will need to be doubled. For example, \\pounds instead of \pounds. If this causes a problem, you can change the CSV separator to a different character using datatooltk's application settings. (This doesn't apply to any of the sample files provided with this book as none of them contain any LATEX commands.)

On page 76 I mentioned that \DTLloaddb and \DTLloadrawdb can't load CSV files where there is an EOL within a cell, but datatooltk can. Let's suppose the file books-multiline.csv contains the following:

```
Title,Price  
"Duck and Goose's  
Adventures",10.00  
"The Return of  
Duck and Goose",11.00
```

This can be converted into a datatool (.dbtex) file using one of the following methods:

1. Run datatooltk in batch mode from a command prompt:

2.2 Loading Data

```
datatooltk --csv books-multiline.csv  
--output books.dbtex
```

Shell

The database label by default is taken from the **CSV** filename. If you want to change it you can use the **--name <db-name>** option:

```
datatooltk --name products2014  
--csv books-multiline.csv --output books.dbtex
```

Shell

If you want to convert **special characters** to commands (using the mapping given in [Table 2.1](#)) you need to use the **--map-tex-specials** option:

```
datatooltk --name products2014  
--csv books-multiline.csv  
--output books.dbtex --map-tex-specials
```

Shell

2. Run **datatooltk** in **graphical user interface (GUI)** mode. If you have installed **datatooltk** on Windows there should be an entry

2.2 Loading Data

in your Start menu that will do this. Otherwise you can run the command `datatooltk-gui`. This should display the window shown in [Figure 2.1](#).

If your [CSV](#) file contains [special characters](#) that you want converted to [L^AT_EX](#) commands, you need to switch on the map T_EX characters via the “TeX” tab in the “Preferences” dialog. (This can be opened using the menu item [Edit→Edit Preferences](#).)

Next use the [File→Import→Import CSV](#) menu item. This will open the dialog box shown in [Figure 2.2](#). Select the CSV file and click on “Import”. The data should now be visible in the main window as shown in [Figure 2.3](#).

The tab above the data shows the database label (the same as the first mandatory argument of `\DTLloaddb` and `\DTLloadrawdb`). If you want to change the default, you can double-click on the tab which will open a dialog box where you can type in a new label. You can now save the data to a datatool (`.dbtex`) file using the [File→Save As](#) menu item.

If you want to import an Excel `.xls` file to a datatool (`.dbtex`) file, you can use the `--xls` and `--sheet` options to `datatooltk`. The argument

2.2 Loading Data

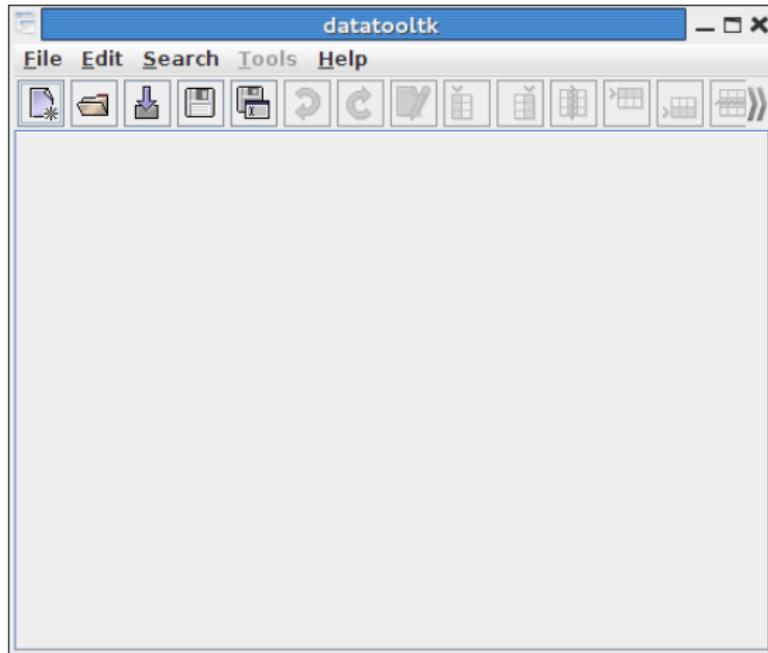


Figure 2.1 datatoolk in Graphical Mode

2.2 Loading Data

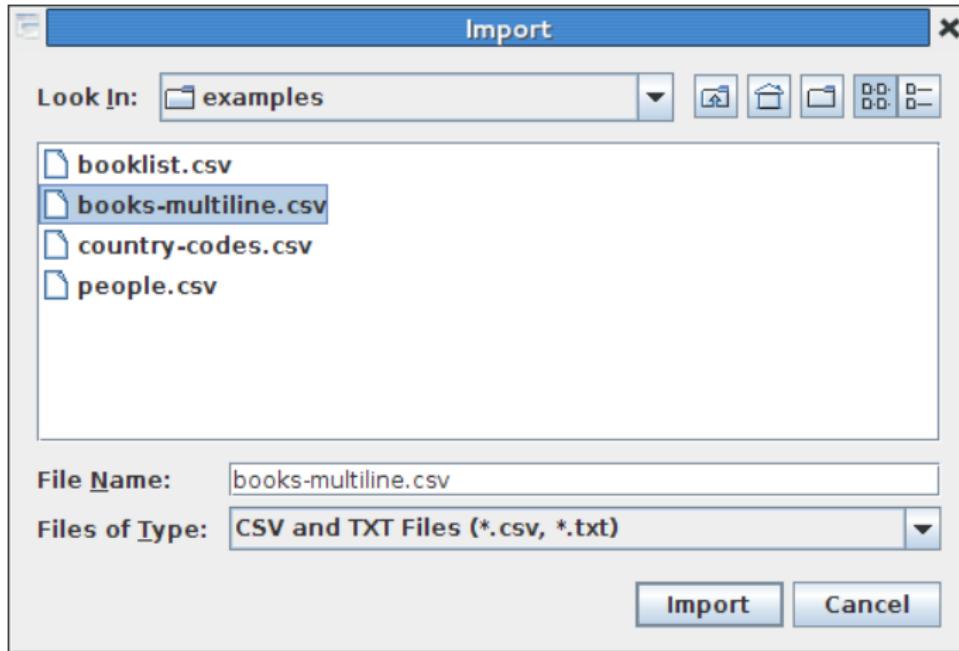


Figure 2.2 Import CSV File

2.2 Loading Data

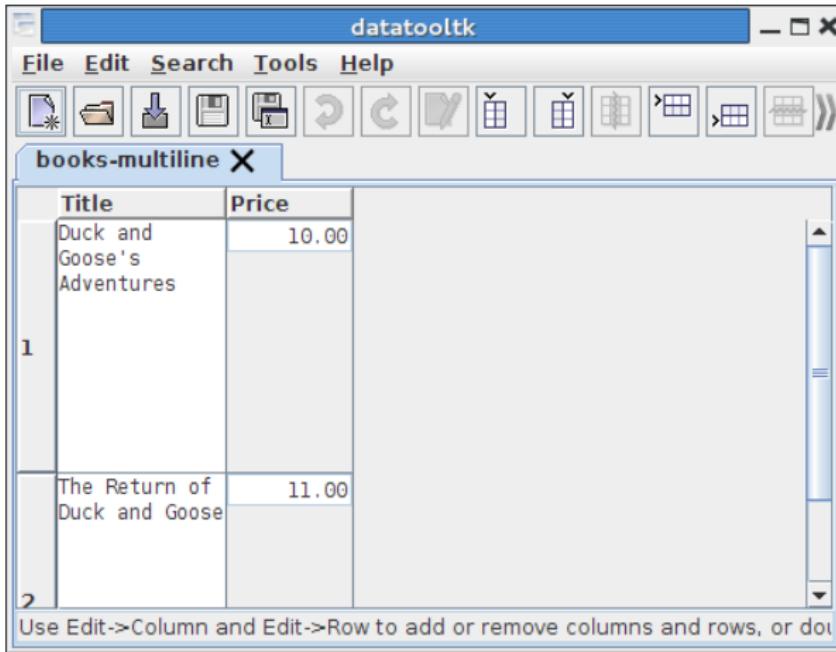


Figure 2.3 Imported CSV Data Shown in Main Window

2.2 Loading Data

of `--xls` is the name of the `.xls` file and the argument of `--sheet` is either an integer (starting from 0) indicating the sheet index or a string identifying the sheet label. Note that no formatting information is read from the Excel file. Any font changes or alignments should be made in your L^AT_EX document. Similarly, you can import an Open Document `.ods` spreadsheet using the `--ods` and `--sheet` options.

EXAMPLE 2. CONVERT AN `.xls` SHEET TO A `.dbtex` FILE

Suppose you have an Excel file called, say, `shop.xls` and it contains two sheets: “products” and “customers”. You can convert, say, the “customers” sheet to a file called `customers.dbtex` using one of the following methods:

1. Run `datatooltk` in batch mode from a command prompt:

```
datatooltk --output customers.dbtex --xls shop.xls  
--sheet customers
```

Shell

Or:

```
datatooltk --output customers.dbtex --xls shop.xls  
--sheet 1
```

Shell

2.2 Loading Data

(The customer data is in the second sheet, but indices start from 0 so it's sheet 1.)

2. Run `datatooltk` in GUI mode using `datatooltk-gui`, as described above, and set the map TeX characters option and header row settings if required.

Next use the File→Import→Import Spreadsheet menu item. This will open a file selector dialog box. Select the .xls file (shop.xls in this example) and click on the “Import” button. Another dialog box will appear in which you need to select the required sheet name (“customers” in this example).

Click on “Okay” and the data will be displayed in the main window. You can then save the data to a .dbtex file using the File→Save As menu item.

Note that formatting information isn't fetched from the spreadsheet. You will have to add any necessary font changing commands when you display the data. This helps to provide a consistent style throughout the document.

You can also use `datatooltk` to import data from a MySQL database. This is slightly more complicated as you need to tell `datatooltk` the data-

2.2 Loading Data

base, the [SQL](#) SELECT statement⁵, the user name and password. (You may also need to change the [SQL](#) port and host, if different from the defaults.)

EXAMPLE 3. IMPORTING SQL DATA TO A .DBTEX FILE

Suppose you have a database called samples and in that database you have a table called books and let's suppose the user name for the samples database is "sampleuser". Then you can import the data into a datatool (.dbtex) file using one of the following methods:

1. Run [datatooltk](#) in batch mode from a command prompt:

```
datatooltk --output books.dbtex --sqluser sampleuser
--sqldb samples --sql "SELECT * FROM books"
```

Shell

This will prompt you for a password from the console. As before, you can use the `--map-tex-specials` option if required.

2. Run [datatooltk](#) in GUI mode using [datatooltk-gui](#), as described above, and set the map TeX characters option if required.

⁵[SQL](#) statements are beyond the scope of this guide, but for further details I recommend you read "Managing & Using MySQL" [71].

2.2 Loading Data

Next use the File→Import→Import SQL menu item. This will open the “Import SQL” dialog box. Edit the SELECT statement as required and enter the database and user name in the appropriate fields. For example, as shown in [Figure 2.4](#).

Then click on “Okay” and enter the password when prompted.

The data should now be visible in the main window as illustrated in [Figure 2.5](#).

3. If you want to change the column header details, double-click on the button at the top of the required column. This will display the dialog box shown in [Figure 2.6](#).
4. Once the data has been fetched, you can save it using File→Save As.

Remember that you can use the SELECT statement to filter unwanted rows, sort data or join the data with data from other tables.

If you use [arara](#) to build your document, you can use the `datatooltk` directive, but you must make sure you have at least v4.0 of arara installed.

2.2 Loading Data

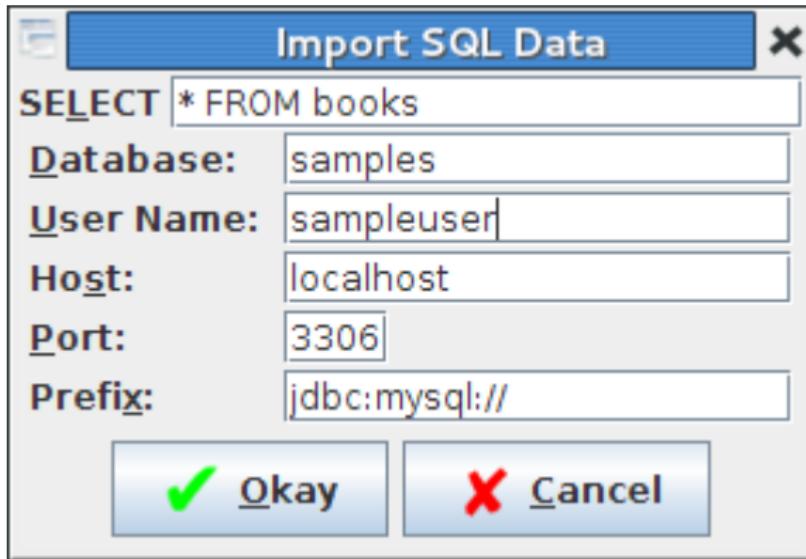


Figure 2.4 Import SQL Data

2.2 Loading Data

The screenshot shows the 'datatooltk' application window. The menu bar includes File, Edit, Search, Tools, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Database management. The main window title is 'books X'. It displays a table with columns: id, title, author, format, and price. The data consists of two rows:

id	title	author	format	price
1	The Adventures of Duck and Goose	Sir Quackalot	paperback	10.99
2	The Return of Duck and Goose	Sir Quackalot	paperback	11.99

At the bottom, a message says: 'Use Edit->Column and Edit->Row to add or remove columns and rows, or do'.

Figure 2.5 Imported SQL Data Shown in Main Window

2.2 Loading Data

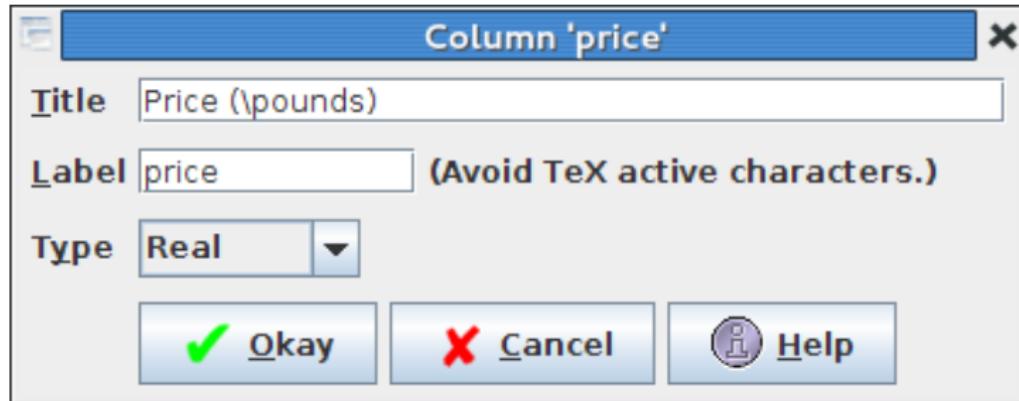


Figure 2.6 Changing the Column Header Text

2.2 Loading Data

EXAMPLES USING ARARA

1. Fetch the data from a CSV file called booklist.csv:

```
% arara: datatooltk: {
% arara: --> output: books.dbtex,
% arara: --> csv: booklist.csv }
% arara: pdflatex
\documentclass{article}

\usepackage{datatool}

\DTLloaddbtx{\books}{books.dbtex}

\begin{document}
% Do stuff with data.
\end{document}
```

↑ Input

↓ Input

Remember that you can also use conditionals to prevent unnecessary application calls. For example:

2.2 Loading Data

↑ Input

```
% arara: datatooltk: {
% arara: --> output: books.dbtex,
% arara: --> csv: booklist.csv }
% arara: --> if changed(toFile("booklist.csv"))
% arara: --> || missing(toFile("books.dbtex"))
% arara: pdflatex while changed("tex")
% arara: --> || changed(toFile("books.dbtex"))
% arara: --> || missing("pdf")
% arara: --> || found("log", "Rerun")
\documentclass{article}

\usepackage{datatool}

\DTLloaddbtx{\books}{books.dbtex}

\begin{document}
% Do stuff with data.
\end{document}
```

↓ Input

2.2 Loading Data

- Fetch the data from the sheet called “products” in an Excel .xls file called shop.xls:

↑ Input

```
% arara: datatooltk: {
% arara: --> output: products.dbtex,
% arara: --> xls: shop.xls,
% arara: --> sheet: products }
% arara: pdflatex
\documentclass{article}

\usepackage{datatool}

\DTLloaddbtx{\products}{products.dbtex}

\begin{document}
% Do stuff with data.
\end{document}
```

↓ Input

2.2 Loading Data

3. Fetch all the data from the table called `books` in a MySQL database called `samples`:

↑ Input

```
% arara: datatooltk: {
% arara: --> output: books.dbtex,
% arara: --> sqldb: samples,
% arara: --> sqluser: sampleuser,
% arara: --> sql: "SELECT * FROM books" }
% arara: pdflatex
\documentclass{article}

\usepackage{datatool}

\DTLloaddbtx{\books}{books.dbtex}

\begin{document}
% Do stuff with data.
\end{document}
```

↓ Input

2.2 Loading Data

In this case, `datatooltk` will prompt you for the MySQL password associated with the specified user name. There's a slight difference between running `datatooltk` directly from a terminal and running it via `arara`. The terminal usually provides a console for `datatooltk` to use to prompt for a password, but with `arara` there's no console so you'll get a dialog box instead (even if you haven't specified the `--verbose` option when you call `arara`). See [Figure 2.7](#). This will also happen if you invoke `datatooltk` from another application, such as TeXworks.

-  If you're thinking of using a conditional in the `arara` directive, there's no platform-independent way of determining if the database has been modified. The way the data is stored on the hard disk varies not only according to the operating system but also with the way the table was created (MyISAM, InnoDB, etc). For example, on my Linux computer I could test if the file `/var/lib/mysql/samples/books.MYD` has changed, but I would need to run `arara` with `sudo` in order to access the file, which is unwise.

2.2 Loading Data



Figure 2.7 Running `datatooltk` via `arara` uses a dialog box to prompt for a MySQL password instead of using a console.

2.3 ■ Security

When you import data from an [SQL](#) database, you need to enter the password associated with the [SQL](#) user. This must be done each time you run [datatooltk](#) with the `--sql` option. While it is possible to specify the password using the `--sqlpassword` option, this is not advisable as it will be visible to anyone who happens to be looking over your shoulder (and will probably be remembered by the shell's history). It's also not advisable to include the password in an [arara](#) directive since it can be read by anyone with access to the document source.

However, regardless of whether or not you use the `--sqlpassword` option, any confidential data that was imported from the [SQL](#) database will be present in the [datatool](#) (`.dbtex`) file, which other users may be able to access and read. If your operating system supports different permissions for owner and others (such as Unix-based systems) you can use the `--owner-only` option when invoking [datatooltk](#) so that the permissions on the `.dbtex` are set to read and write only for the owner. For example:

```
datatooltk --output people.dbtex --sqluser sampleuser
--sqldb samples --sql "SELECT * FROM people" --owner-only
```

Shell

On Unix-like systems this is equivalent to

2.3 Security

```
datatooltk --output people.dbtex --sqluser sampleuser  
--sqldb samples --sql "SELECT * FROM people"  
chmod 600 people.dbtex
```

Shell

This means that only the owner of the file (you, if you ran the `datatooltk` command under your own user name) can read or write the `people.dbtex` file. Any other users with an account on the same machine should not be able to read or write that file. If your operating system doesn't support different owner and other permissions, the `--owner-only` option will have no effect.

If you are using `arara` the directive is:

```
% arara: datatooltk: {  
% arara: --> output: people.dbtex,  
% arara: --> sqluser: sampleuser,  
% arara: --> owneronly: true,  
% arara: --> sqldb: samples,  
% arara: --> sql: "SELECT * FROM people" }
```

↑ Input

↓ Input

2.3 Security

 Even if you take the precaution of setting the permissions on the .dbtex file in this way, if you've used the data in your document, that data is also likely to be present in the resulting PDF file. If this data is sensitive, you also need to consider changing the permissions of the PDF file as well and possibly use something like `pdftk` to add encryption.

If your `SQL` database contains a mixture of private and public data, but you only want to use the public data in your document, make sure you omit the private data in your `SELECT` statement. For example, instead of using `SELECT *` to fetch all columns replace the asterisk with just the columns you want to fetch. Perhaps you only want to list the countries where you have customers, then you'd just use `SELECT country FROM people` which would only write the country information to the .dbtex file and not the individual customer details.

Remember that `TeX` discards comments in the form

`% <comment text>`

Input

(unless the `category code` of `%` is changed, for example, by the `verbatim` environment). Here the comment text doesn't get hidden or embedded within the resulting PDF, but in other cases don't assume that just because you can't see the text you've included in your source code it won't be somehow present in the PDF file. For example

2.3 Security

```
\textcolor{white}{some text} Input
```

If this text is placed on a white background, it won't be visible to the human eye, but the text will be present in the PDF file and can be read by an electronic device such as a screen reader or it can be extracted using a PDF to text tool. The `censor` package described in §6.4.1 replaces the redacted text with a black rectangle rather than simply obscuring the text by painting a black rectangle on top of it, which means that the redacted text can't be extracted from the PDF.

Unlike Word [89], L^AT_EX doesn't automatically embed private information or revision logs in the document. With L^AT_EX you need to explicitly indicate the author or authors. This is done through the `\author` command in the standard classes. PDFL^AT_EX allows you to add metadata, but again this relies on you explicitly providing the author data. (The `hyperref` package provides a convenient key to do this called `pdfauthor`, which can be passed as a package option or through `\hypersetup{\langle options \rangle}`). It's possible that other classes or packages may use `\author` or provide a similar command to add the author's name to the metadata, but it's still information provided by you and not automatically picked up from your operating system's environment variables or settings.)

T_EX has a `shell escape` mechanism that allows processes to be spawned during the document build. However, since this can be exploited by malicious code, it's usually disabled completely or just enabled in restricted

2.3 Security

mode, which only allows trusted applications (such as `makeindex` or `bibtex`) to be run. You can check the mode on your system by inspecting the log file. For example, if the log file contains “restricted \write18 enabled” near the start, then `TeX` was run in restricted mode. The `shell escape` can be enabled using `TeX`’s `-shell-escape` option, for example:

```
pdflatex -shell-escape myDoc
```

Shell

or disabled using the `-no-shell-escape` option, for example:

```
pdflatex -no-shell-escape myDoc
```

Shell

or the restricted mode can be enabled using `-shell-restricted`, for example:

```
pdflatex -shell-restricted myDoc
```

Shell

Note that `-shell-escape` is normally disallowed.

JavaScript code can be embedded in PDF files, which may be a security issue. The code is run by the PDF viewer when viewing the PDF file, not during the document build. If this is a concern for you, disable this feature in your PDF viewer. As with the author metadata, `TeX` doesn’t automatically embed JavaScript code in the PDF file. You can explicitly embed JavaScript

2.3 Security

code into interactive PDF elements, but obviously don't do this if you don't understand the code you're trying to embed.

You have control over the commands you enter into your source code, but what about the document class or packages that you load? If you have the `shell escape` disabled or restricted, then this automatically prevents any class or package from running dangerous applications. Some classes or packages may embed JavaScript code if their purpose is to create an interactive PDF document. If this concerns you, disable JavaScript in your PDF viewer, as mentioned above. Remember also that class and package files can be viewed in any text editor, so it's possible to inspect the code. While this may not appeal to you, if it's a well-used class or package that's on both MiK^TE_X and T_EX Live, then the chances are that someone else already has.

What about applications such as `arara` and `lutexmk`? You can add code to `lutexmk` through the `.lutexmkrc` file, but if you're writing the code, you should know what that code does. It's possible to embed Java code within the conditional part of `arara` directives, but again if you are writing the code it's up to you to ensure you don't write anything dangerous. If someone else has supplied a L^AT_EX document that contains `arara` directives, you can search for them in the document source file before running `arara` on it. However `arara` will only execute a directive that has a corresponding `.yaml` file in its rules directory, and you can also use `--dry-run` to see what

2.4 Sorting Data

commands would be executed.

Remember that if the application is available on TeX Live, then it has to have the source available. In fact, since `latexmk` is a Perl script, you can open it in a text editor. The `arara` directives are all defined in `.yaml` files, which again can be viewed in a text editor. An active open source community with a central file repository is far more likely to detect and flag malicious code than any users of proprietary systems.

But what if someone accesses your computer and modifies the `.latexmkrc` or `.yaml` files? In that case, you have far more to worry about than the integrity of your TeX distribution.

Other security issues you might want to consider are discussed in [§6.4](#) and [§13 Collaborating on Documents](#).

2.4 ■ Sorting Data

Once you've loaded your data you can sort it using:

`\dtlsort[<replacement list>]{<criteria>}{<db-name>}{<handler>}`

Definition

where `<db-name>` is the label that identifies the database. The `<criteria>` argument is a [comma-separated list](#) of column labels that indicate the sort order. For example, if you first want to sort on the `surname` column and

2.4 Sorting Data

then on the forenames column the `<criteria>` should be `surname,forenames` (make sure you don't have any unwanted spaces in the list). You can optionally add `=<order>` after the column label where `<order>` is either ascending or descending. If omitted, ascending is assumed. For example, to sort in descending order, first by surname and then by forenames, the `<criteria>` should be:

`surname=descending,forenames=descending`

The `<handler>` argument is a control sequence that's used for the comparisons. The `datatool` package comes with four handlers:

1. A case-sensitive comparison:

`\dtlcompare`

Definition

2. A case-insensitive comparison:

`\dtlicompare`

Definition

3. English word-ordering comparison (as described by the Oxford Style Manual [74]):

2.4 Sorting Data

\dtlwordindexcompare

[Definition](#)

4. English letter-ordering comparison:

\dtlletterindexcompare

[Definition](#)

The last two are intended for indexes. If you want any further details about those handlers, see the datatool user guide [95]. The first two handlers, \dtlcompare and \dtlicompare, are the ones you're most likely to need for administrative purposes. The datatool package provides convenient shortcuts:

\DTLsort[⟨replacement list⟩]{⟨criteria⟩}{⟨db-name⟩}

[Definition](#)

which uses \dtlcompare and

\DTLsort*[⟨replacement list⟩]{⟨criteria⟩}{⟨db-name⟩}

[Definition](#)

which uses \dtlicompare.

The optional argument ⟨replacement list⟩ is provided in case null values are encountered. You're unlikely to have null values if you load your data from a CSV file, but you may have null values if you use datatooltk to fetch data from a SQL database.

2.4 Sorting Data

For example, suppose you want to sort your data according to the author column, but if there's a null value in that column then sort it by the title column, you would need to do

```
\DTLsort[title]{author}{books}
```

Input

(where books is the label identifying your data.)

 T_EX isn't designed for data analysis, so sorting your data within your document in this way isn't very efficient. It's therefore better to sort your data before you load it in your document. This can be done using **datatooltk**'s --sort option (in batch mode) or via the Tools→Sort menu item (in GUI mode). Alternatively, if you're fetching data from a SQL database, it's more efficient to append the ORDER BY statement to your SELECT statement.

EXAMPLES:

1. Fetch and sort data from a CSV file.

- Either:

```
\DTLloaddb{books}{booklist.csv}
```

↑ Input

2.4 Sorting Data

```
\DTLsort{author}{books}
```

↓ Input

- Or use `datatooltk`:

```
datatooltk --output books.dbtex  
--csv booklist.csv --sort author
```

Shell

and then in your document:

```
\DTLloaddbtx{\books}{books.dbtex}
```

Input

2. Fetch and sort data from a `SQL` database:

```
datatooltk --output books.dbtex --sqluser sampleuser  
--sqldb samples --sql "SELECT * FROM books ORDER BY  
author"
```

Shell

and then in your document:

```
\DTLloaddbtx{\books}{books.dbtex}
```

Input

2.5 Sample Data

This section describes the sample [CSV](#) files, Excel spreadsheet and [SQL](#) database tables that will be used in some of the examples and exercises throughout the rest of this book. This includes sample data regarding hypothetical people. These people could represent, for example, students or customers or members of an organisation. This leads on to a couple of points that aren't specific to \TeX , but are of a more general nature.

If you intend to store personal data, make sure you are aware of your country's data storage legislation. For example, in the United Kingdom you need to be familiar with the data protection act and you will probably also need to register as a data controller with the [Information Commissioner's Office \(ICO\)](#), although there are some exemptions. (At the time of writing it usually costs £35 per year, but may cost £500 per year, depending on your organisation's size.) If you are an employee, check with your company's administrative office for further details.

The sample data includes both a gender and a title field. The gender is usually not required, unless you need to know whether to use male or female pronouns (for example, "him" or "her") or where the gender has some significance (for example, in medical data). Some people view a request for their gender to be intrusive, but if it is genuinely needed, make sure you include it in the data rather than omitting it and trying to de-

2.5 Sample Data

termine the gender from the person's name. Additionally, I recommend you don't use a gender field as a replacement for the title field. It's not advisable to assume "Mr" or "Ms" on the basis of gender. People's preferences are varied. Some don't like the formality of titles at all, or simply don't care how they're addressed, but some women object to being addressed as "Ms". While it's true that some professional women prefer "Ms" as they feel their marital status is no one's business, there are, on the other hand, some women with a professional title, such as "Dr" or "Prof", who object to being "demoted" to a "Ms", particularly if they work in a male-dominated environment and they perceive, rightly or wrongly, that their male colleagues are being shown more respect professionally. There are, of course, plenty of other titles as well, such as "Rev", so I think it's best to find out how people prefer to be addressed. If the data is collected by a third party (such as an online store script) and it doesn't provide you with the person's title, you may want to consider just addressing individuals by both their forename and surname (such as "Dear John Smith") rather than guessing a title to avoid unwittingly causing offence. (Remember there are also people, of certain cultures or age, who object to the informality of being addressed by their first name.)

2.5 Sample Data

2.5.1 Sample CSV Files

Some of these files contains [UTF-8](#) characters, so you'll also need to use:

```
\usepackage[utf8]{inputenc}  
\usepackage[T1]{fontenc}
```

 Input

 Input

(or just `fontspec` [76] for \LaTeX) and make sure your text editor is set to [UTF-8](#) encoding. The sample [CSV](#) files are as follows (you can download them from [the examples page](#)):

1. `booklist.csv` ([download](#))

This is a list of sample book titles. They could represent, say, products in a shop or a reading list for students. The `id` column provides a number that uniquely identifies the book. (This would typically be an ISBN in a real life book list or the ISBN might be in an additional column.)

```
id,title,author,format,price  
1,"The Adventures of Duck and Goose","Sir Quackalot",paperback,  
10.99
```



2.5 Sample Data

2,"The Return of Duck and Goose","Sir Quackalot",paperback,	↔
19.99	
3,"More Fun with Duck and Goose","Sir Quackalot",paperback,	↔
12.99	
4,"Duck and Goose on Holiday","Sir Quackalot",paperback,	↔
11.99	
5,"The Return of Duck and Goose","Sir Quackalot",hardback,	↔
19.99	
6,"The Adventures of Duck and Goose","Sir Quackalot",hardback,	↔
18.99	
7,"My Friend is a Duck","A. Parrot",paperback,14.99	
8,"Annotated Notes on the 'Duck and Goose' chronicles",	↔
"Prof Macaw",ebook,8.99	
9,"'Duck and Goose' Cheat Sheet for Students",	↔
"Polly Parrot",ebook,5.99	
10,"'Duck and Goose': an allegory for modern times?",	↔
"Bor Ing",hardback,59.99	

This file can be loaded using:

2.5 Sample Data

\DTLloaddb{books}{booklist.csv}

Input

2. people.csv ([download](#))

This is a list of sample people. They could represent, say, customers or students or members of an organization. The `id` column provides a number that uniquely identifies the person. (For example, a student's registration number.) The `subscribed` field indicates whether the person wants to be subscribed to some form of mailing list where a value of 1 means they want to be subscribed. The `gender` column can be either "m" (male) or "f" (female). The `dob` column is the date of birth in the ISO format `<year>-<month>-<day>`.

```
id,forenames,surname,title,address1,address2,town,county,  
country,postcode,subscribed,gender,dob  
1,Polly,Parrot,Miss,42 The Lane,,Some Town,Noshire,gb,  
AB1 2XY,1,f,1970-12-31  
2,Mabel,Canary,Mrs,24 The Street,Some Village,Some Town,  
Noshire,gb,AB1 2YZ,0,f,1968-01-23  
3,Zoe,Zebra,Ms,856 The Avenue,,Some City,CA,us,123456,1,  
f,1989-07-16
```

2.5 Sample Data

```
4,José,Arara,,Nenhuma Rua,,São Paulo,,br,123457,1,m,  
1991-05-30  
5,Dickie,Duck,Mr,1 The Street,Another Village,Some City,  
Imagineshire,gb,YZ1 2AB,0,m,1952-11-25  
6,Fred,Canary,Mr,24 The Street,Some Village,Some Town,  
Noshire,gb,AB1 2YZ,1,m,1967-08-04
```

This file can be loaded using:

```
\DTLloaddb{people}{people.csv}
```

Input

3. country-codes.csv ([download](#))

This is a list of country codes. It contains 250 lines. You can download the entire file from [the examples page](#). For brevity, only the header row and the rows containing codes that are referenced in `people.csv` are shown below:

```
code,name  
br,Brazil  
gb,United Kingdom  
us,United States
```

2.5 Sample Data

This file can be loaded using:

```
\DTLloaddb{countries}{country-codes.csv}
```

Input

4. ordergroups.csv ([download](#))

This is a list of customer orders. The first column is a number that uniquely identifies the order, the second column is a cross-reference to the id field in the people.csv file above, indicating the purchaser, the third column is a discount applied to the order, and the fourth column is the cost of postage and packaging.

```
id,customerid,discount,postage
1,2,0,5.00
2,4,2.50,20.00
3,1,0,5.00
```

(In practice, this would typically have extra fields, such as the dispatch status and order date.)

This file can be loaded using:

2.5 Sample Data

\DTLloaddb{ordergroups}{ordergroups.csv}

Input

5. orders.csv ([download](#))

This is a list of partial customer orders. Each row represents part of an order for one of the book titles listed in booklist.csv by one of the people listed in people.csv. There are four columns: the first is a number that uniquely identifies each order part, the second is a number that identifies the entire order (matching the id field in ordergroups.csv), the third column is a cross-reference to the id field in the booklist.csv file above, and the fourth column is the quantity ordered.

```
id,groupid,bookid,quantity
1,1,6,1
2,1,1,4
3,2,10,1
4,2,7,20
5,2,8,1
6,3,1,4
7,3,6,5
8,3,7,2
```

2.5 Sample Data

The first two rows form a single order (id 1 from the `ordergroups.csv` file). The order consists of two parts: one copy of the book whose id is 6 (the hardback version of "The Adventures of Duck and Goose") and four copies of the book whose id is 1 (the paperback version of "The Adventures of Duck and Goose"). The next three rows form another order (id 2 from `ordergroups.csv`). The order consists of three parts: one copy of the book whose id is 10, twenty copies of the book whose id is 7 and one copy of the book whose id is 8. The last four rows form the third order (id 3 from `ordergroups.csv`). The order consists of three parts: four copies of the book whose id is 1, five copies of the book whose id is 6 and two copies of the book whose id is 7.

This file can be loaded using:

```
\DTLloaddb{orders}{orders.csv}
```

Input

2.5.2 Sample XLS File

Most of the examples in this book will use data from either [CSV](#) files or [SQL](#) tables, but [datatooltk](#) can also import from Excel .xls files so there is one sample file, `shop.xls`, to illustrate this. The `shop.xls` file has

2.5 Sample Data

two sheets called “products” and “customers”. The first sheet is shown in [Figure 2.8](#). This has three columns. The second and third columns (“B” and “C”) have been set to a currency format. In addition, the third column contains formulae instead of an explicitly entered number.

The data can be fetched via [datatooltk](#). For example, to fetch the data from the first sheet (“products”):

```
datatooltk --output products.dbtex --xls shop.xls  
--sheet products
```

Shell

Or using an [arara](#) directive:

```
% arara: datatooltk: {  
% arara: --> output: products.dbtex,  
% arara: --> xls: shop.xls,  
% arara: --> sheet: products }
```

↑ Input

↓ Input

The resulting `products.dbtex` file can be loaded in your document using:

2.5 Sample Data

The screenshot shows a LibreOffice Calc spreadsheet titled "shop.xls". The interface includes a menu bar with File, Edit, View, Insert, Format, Tools, Data, and Window. Below the menu is a toolbar with various icons for file operations like Open, Save, Print, and a search function. The formula bar at the top shows the cell reference C5 and the formula =B5*1.2.

	A	B	C	D	E
1	Product	Price (ex VAT)	Price (inc VAT)		
2	Ink cartridge	£25.00	£30.00		
3	Mouse mat	£12.00	£14.40		
4	USB stick	£15.00	£18.00		
5	Pen	£2.50	£3.00		
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

At the bottom, there are tabs for "products" and "customers", and a status bar showing "Sheet 1 / 2", "PageStyle...products", "Sum=£3.00", and navigation icons.

Figure 2.8 Sample XLS File (First Sheet)

2.5 Sample Data

```
\DTLloaddbtx{\products}{products.dbtex}
```

Input

You can also import Open Document .ods spreadsheets using a similar method to the above but you need to use --ods instead of --xls.

2.5.3 ■ Sample SQL Tables

The sample [SQL](#) database called “samples” is created using:

```
CREATE DATABASE samples;
```

A sample user called “sampleuser” with the password “sample-passwd” is created using:

```
GRANT SELECT ON samples.* TO 'sampleuser'@'localhost'  
IDENTIFIED BY 'sample-passwd';
```

Remember to switch to this database before you try creating the tables in it:

```
USE samples;
```

If you like, you can [download](#) samples.sql and add this sample database (including the tables below) using:

2.5 Sample Data

```
mysql -u root -p < samples.sql
```

Shell

(Alternatively, you can use one of the [GUI](#) MySQL tools, such as [MySQL Workbench](#).)

The tables included in this sample database are analogous to the [CSV](#) files described in §2.5.1. As with those files, some of the [SQL](#) tables include [UTF-8](#) characters, so you'll also need to use:

```
\usepackage[utf8]{inputenc}  
\usepackage[T1]{fontenc}
```

↑ Input

↓ Input

(or just [fontspec](#) [76] for X_EL^AT_EX) and make sure your text editor is set to [UTF-8](#) encoding.

The tables are defined as follows:

- books

This table is created using:

```
CREATE TABLE books  
(
```

2.5 Sample Data

```
id INT PRIMARY KEY AUTO_INCREMENT,  
title VARCHAR(64),  
author VARCHAR(32),  
format ENUM('paperback', 'hardback', 'ebook'),  
price DECIMAL(5,2)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

The data is added to the table via:

```
INSERT INTO books (title, author, format, price)  
VALUES ('The Adventures of Duck and Goose',  
'Sir Quackalot', 'paperback', 10.99);  
INSERT INTO books (title, author, format, price)  
VALUES ('The Return of Duck and Goose', 'Sir Quackalot',  
'paperback', 11.99);  
INSERT INTO books (title, author, format, price)  
VALUES ('More Fun with Duck and Goose', 'Sir Quackalot',  
'paperback', 12.99);  
INSERT INTO books (title, author, format, price)  
VALUES ('Duck and Goose on Holiday', 'Sir Quackalot',  
'paperback', 11.99);  
INSERT INTO books (title, author, format, price)  
VALUES ('The Return of Duck and Goose', 'Sir Quackalot',
```

2.5 Sample Data

```
'hardback', 19.99);
INSERT INTO books (title, author, format, price)
VALUES ('The Adventures of Duck and Goose', 'Sir Quackalot',
'hardback', 18.99);
INSERT INTO books (title, author, format, price)
VALUES ('My Friend is a Duck', 'A. Parrot', 'paperback',
14.99);
INSERT INTO books (title, author, format, price)
VALUES (
'Annotated Notes on the \'Duck and Goose\' chronicles',
'Prof Macaw', 'ebook', 8.99);
INSERT INTO books (title, author, format, price)
VALUES ('\'Duck and Goose\' Cheat Sheet for Students',
'Polly Parrot', 'ebook', 5.99);
INSERT INTO books (title, author, format, price)
VALUES ('\'Duck and Goose\': an allegory for modern times?',
'Bor Ing', 'hardback', 59.99);
```

2. people

This table is created using:

```
CREATE TABLE people
```

2.5 Sample Data

```
(  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    forenames VARCHAR(32) NOT NULL,  
    surname VARCHAR(32) NOT NULL,  
    title VARCHAR(8),  
    address1 VARCHAR(32) NOT NULL,  
    address2 VARCHAR(32),  
    town VARCHAR(32) NOT NULL,  
    county VARCHAR(32),  
    country CHAR(2) NOT NULL,  
    postcode VARCHAR(32),  
    subscribed BIT(1) DEFAULT 0,  
    gender ENUM('male', 'female'),  
    dob DATE,  
    INDEX(id, surname)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

The data is added to the table via:

```
INSERT INTO people  
(forenames, surname, title, address1, address2,  
town, county, country, postcode, subscribed, gender, dob)  
VALUES ('Polly', 'Parrot', 'Miss', '42 The Lane', NULL,
```

2.5 Sample Data

```
'Some Town', 'Noshire', 'gb', 'AB1 2XY', 1, 'female',
'1970-12-31');

INSERT INTO people
(forenames, surname, title, address1, address2,
town, county, country, postcode, subscribed, gender, dob)
VALUES ('Mabel', 'Canary', 'Mrs', '24 The Street',
'Some Village', 'Some Town', 'Noshire', 'gb', 'AB1 2YZ',
0, 'female', '1968-01-23');

INSERT INTO people
(forenames, surname, title, address1, address2,
town, county, country, postcode, subscribed, gender, dob)
VALUES ('Zöe', 'Zebra', 'Ms', '856 The Avenue', NULL,
'Some City', 'CA', 'us', '123456', 1, 'female', '1989-07-16');

INSERT INTO people
(forenames, surname, title, address1, address2,
town, county, country, postcode, subscribed, gender, dob)
VALUES ('José', 'Arara', NULL, 'Nenhuma Rua', NULL,
'São Paulo', NULL, 'br', '123457', 1, 'male', '1991-05-30');

INSERT INTO people
(forenames, surname, title, address1, address2,
town, county, country, postcode, subscribed, gender, dob)
VALUES ('Dickie', 'Duck', 'Mr', '1 The Street',
```

2.5 Sample Data

```
'Another Village', 'Some City', 'Imagineshire', 'gb',
'YZ1 2AB', 0, 'male', '1952-11-25');

INSERT INTO people
(forenames, surname, title, address1, address2,
town, county, country, postcode, subscribed, gender, dob)
VALUES ('Fred', 'Canary', 'Mr', '24 The Street',
'Some Village', 'Some Town', 'Noshire', 'gb', 'AB1 2YZ',
1, 'male', '1967-08-04');
```

3. countries

This is a list of country codes. The table is created using:

```
CREATE TABLE countries
(
    code CHAR(2) PRIMARY KEY NOT NULL,
    name VARCHAR(64) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

This table contains 249 entries. For brevity, only the rows that contain codes referenced in the people table are shown below:

```
INSERT INTO countries (code, name)
```

2.5 Sample Data

```
VALUES ('br', 'Brazil');
INSERT INTO countries (code, name)
VALUES ('gb', 'United Kingdom');
INSERT INTO countries (code, name)
VALUES ('us', 'United States');
```

You can view or download the complete list from the [examples web page](#).

4. `ordergroups` This is a list of order groups. The table is created using:

```
CREATE TABLE ordergroups
(
    id INT PRIMARY KEY AUTO_INCREMENT,
    customerid INT NOT NULL REFERENCES people (id),
    discount DECIMAL(5,2),
    postage DECIMAL(5,2)
) ENGINE=MyISAM;
```

The data is added to the table via:

```
INSERT INTO ordergroups (customerid, discount, postage)
```

2.5 Sample Data

```
VALUES (2, 0.0, 5.0);
INSERT INTO ordergroups (customerid, discount, postage)
VALUES (4, 2.5, 20.0);
INSERT INTO ordergroups (customerid, discount, postage)
VALUES (1, 0.0, 5.0);
```

5. orders

This is a list of partial orders. The table is created using:

```
CREATE TABLE orders
(
    id INT PRIMARY KEY AUTO_INCREMENT,
    groupid INT NOT NULL REFERENCES ordergroups (id),
    bookid INT NOT NULL REFERENCES books (id),
    quantity INT NOT NULL
) ENGINE=MyISAM;
```

The data is added to the table via:

```
INSERT INTO orders (groupid, bookid, quantity)
VALUES (1, 6, 1);
INSERT INTO orders (groupid, bookid, quantity)
```

2.5 Sample Data

```
VALUES (1, 1, 4);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (2, 10, 1);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (2, 7, 20);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (2, 8, 1);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (3, 1, 4);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (3, 6, 5);
INSERT INTO orders (groupid, bookid, quantity)
VALUES (3, 7, 2);
```

This data can be fetched via `datatooltk`. For example, to fetch the data from the books table:

```
datatooltk --output books.dbtex --sqluser sampleuser
--sqldb samples --sql "SELECT * FROM books"
```

Shell

If you prefer to use `arara` you can use

2.5 Sample Data

```
% arara: datatooltk: {
% arara: --> output: books.dbtex,
% arara: --> sqluser: sampleuser,
% arara: --> sqldb: samples,
% arara: --> sql: "SELECT * FROM books" }
```

↑ Input

↓ Input

The resulting file `books.dbtex` can now be loaded in your document via:

```
\DTLloaddbtex{\books}{books.dbtex}
```

Input

If required, the column headers can be set using `\DTLsetheader`. For example:

```
\DTLsetheader{\books}{id}{ID}
\DTLsetheader{\books}{title}{Title}
\DTLsetheader{\books}{author}{Author}
\DTLsetheader{\books}{format}{Format}
\DTLsetheader{\books}{price}{Price (\pounds)}
```

↑ Input

↓ Input

2.6 Displaying Tabulated Data

Remember you can use the SELECT statement to sort or filter the data or join with another table. For example, if you only want to fetch customers in the UK and order them by their surname:

```
datatooltk --output people.dbtex --sqluser sampleuser
--sqldb samples --sql "SELECT * FROM people
WHERE country='gb' ORDER BY surname"
```

Shell

2.6 □ Displaying Tabulated Data

Once you have [loaded your data](#) you can display it using:

`\DTLdisplaydb[omit list]{{db-name}}`

Definition

where *<db-name>* is the database label. The optional argument *(omit list)* is a [comma-separated list](#) of labels indicating which columns you want omitted from the [tabular](#) environment. Make sure you don't have any unwanted spaces in *(omit list)*.

The `\DTLdisplaydb` command uses a [tabular](#) environment internally and should typically go inside a [table](#) environment. If the data is too big to fit on one page, you can instead use:

2.6 Displaying Tabulated Data

`\DTLdisplaylongdb[⟨options⟩]{⟨db-name⟩}`

Definition

This uses the `longtable` environment defined by the `longtable` package [11] instead of the `tabular` environment (see §4.3). As with `\DTLdisplaydb`, `⟨db-name⟩` indicates the label identifying the data. Note that `\DTLdisplaylongdb` should not be put inside a `table` environment.

The optional argument `⟨options⟩` is a `key=value list`. The following keys are available:

`caption` The caption for the `longtable`.

`contcaption` The continuation caption for the `longtable`.

`shortcaption` The caption to be used in the list of tables.

`label` The label for this table. (To be used in `\ref{⟨label⟩}`, if required.)

`omit` Comma-separated list of labels identifying columns to be omitted.

`foot` The `longtable`'s foot.

`lastfoot` The foot for the last page of the `longtable`.

Remember to load the `longtable` package if you want to use `\DTLdisplaylongdb`.

2.6 Displaying Tabulated Data

EXAMPLE 4. DISPLAY PRODUCT LIST

Suppose I want to display the data from my sample `booklist.csv` file in a `table` environment, but I don't want to include the `id` or `author` columns:

```
% Load "booklist.csv":  
\DTLloaddb  
[headers={Id,Title,Author,Format,Price (\pounds)}] % column headers  
{books}{booklist.csv}  
  
\begin{table}[htbp]  
\caption{Products}  
\label{tab:products}  
\centering  
\DTLdisplaydb[id,author]{books}  
\end{table}
```

↑ Input

↓ Input

This produces [Table 2.2](#).

Suppose instead I want to use the equivalent `books` SQL table. If I'm not interested in using the `id` or `format` columns at all in the document, I can exclude them when I'm importing the data. From the command line:

2.6 Displaying Tabulated Data

Table 2.2 Products

Title	Format	Price (£)
The Adventures of Duck and Goose	paperback	10.99
The Return of Duck and Goose	paperback	11.99
More Fun with Duck and Goose	paperback	12.99
Duck and Goose on Holiday	paperback	11.99
The Return of Duck and Goose	hardback	19.99
The Adventures of Duck and Goose	hardback	18.99
My Friend is a Duck	paperback	14.99
Annotated Notes on the 'Duck and Goose' chronicles	ebook	8.99
'Duck and Goose' Cheat Sheet for Students	ebook	5.99
'Duck and Goose': an allegory for modern times?	hardback	59.99

2.6 Displaying Tabulated Data

```
datatooltk --output books.dbtex --sqldb samples  
--sqluser sampleuser --sql "SELECT title, author, price  
FROM books"
```

Shell

or using an `arara` directive:

```
% arara: datatooltk: {  
% arara: --> output: books.dbtex,  
% arara: --> sqldb: samples,  
% arara: --> sqluser: sampleuser,  
% arara: --> sql: "SELECT title, author, price FROM books" }
```

↑ Input

↓ Input

Once the file `books.dbtex` has been created I can load it into my document using:

```
\DTLloaddbtex{\books}{books.dbtex}
```

Input

and set the headers using:

2.6 Displaying Tabulated Data

↑ Input

```
\DTLsetheader{\books}{title}{Title}
\DTLsetheader{\books}{author}{Author}
\DTLsetheader{\books}{format}{Format}
\DTLsetheader{\books}{price}{Price (\pounds)}
```

↓ Input

(Alternatively, if I import the [SQL](#) data using [datatooltk-gui](#) I can set the header text by double-clicking on the column header as described in [§2.2.2.](#))

The table can now be created in the same way as before. Putting it all together:

↑ Input

```
% Load "books.dbtex":
\DTLloaddbtex{\books}{books.dbtex}

% Set the headers:
\DTLsetheader{\books}{title}{Title}
\DTLsetheader{\books}{author}{Author}
\DTLsetheader{\books}{format}{Format}
```

2.6 Displaying Tabulated Data

```
\DTLsetheader{\books}{price}{Price (\pounds)}
```

% Display the data:

```
\begin{table}[htbp]
\caption{Products}
\label{tab:products}
\centering
\DTLdisplaydb[id,author]{books}
\end{table}
```

↓ Input

This produces the same result shown in [Table 2.2](#).

A longer database, such as the sample `country-codes.csv` file, would need `\DTLdisplaylongdb`, which requires the `longtable` package:

```
\usepackage{longtable}
\usepackage{datatool}
```

↑ Input

↓ Input

Now load the data:

2.6 Displaying Tabulated Data

```
\DTLloaddb{countries}{country-codes.csv}
```

Input

Later in the document, display the data in a `longtable` environment:

```
\DTLdisplaylongdb
[
  caption={A Sample Long Table},% main caption
  contcaption={A Sample Long Table (Continued)},% continuation
  label={tab:countries},% table label
  foot={\emph{Continued on next page}},% table foot
  lastfoot={}% final table foot
]
{countries}
```

↑ Input

↓ Input

The table can then be referenced using:

```
\label{tab:countries}
```

Input

You can [download](#) or [view](#) the full example document using [CSV](#) files,
or [download](#) or [view](#) the example using [SQL](#) database.

2.6 Displaying Tabulated Data

The commands `\DTLdisplaydb` and `\DTLdisplaylongdb` are useful if you just want to list the data, but if you want to modify the displayed format (for example, if you want to swap columns or highlight a row) then you'll need to construct the contents of the `tabular` or `longtable` environments by iterating through the data using `\DTLforeach`, described in §2.7.1.

 Take care if you have imported your data from an Excel .xls file as the formatting information isn't imported. This may cause an unexpected result when you use `\DTLdisplaydb` or `\DTLdisplaylongdb`. This is illustrated in the example below.

EXAMPLE 5. DISPLAYING DATA IMPORTED FROM A SPREADSHEET

Recall the spreadsheet shown in Figure 2.8. Viewed in a spreadsheet application, the second and third columns are displayed as currency with two decimal places. Now suppose I import the data from the “products” sheet using `datatooltk`:

```
datatooltk --output shop-products.dbtex --xls shop.xls  
--sheet products
```

Shell

and load the resulting `shop-products.dbtex` file in my document:

```
\DTLloaddbtex{\xlsproducts}{shop-products.dbtex}
```

Input

then

2.6 Displaying Tabulated Data

↑ Input

```
\begin{table}
  \caption{Data imported from \texttt{shop.xls}}
  \label{tab:xlsproducts}
  \centering
  \DTLdisplaydb{\xlsproducts}
\end{table}
```

↓ Input

produces [Table 2.3](#). Note that the second and third columns are no longer displayed as currency nor are the numbers rounded to two decimal places. (See [Example 6](#) for a different approach that rounds and formats the price columns.)

Table 2.3 Data Imported From `shop.xls`

Product	Price (ex VAT)	Price (inc VAT)
Ink cartridge	25.0	30.0
Mouse mat	12.0	14.399999999999999
USB stick	15.0	18.0
Pen	2.5	3.0

2.6 Displaying Tabulated Data

You can [download](#) or [view](#) this example document.

EXERCISE 1. LOADING AND DISPLAYING DATA

Try loading data from the sample `booklist.csv` file or from the `books` SQL table and displaying it in a `table`. Then try displaying the sample `country-codes.csv` file or `countries` SQL table data in a `longtable`.

FOR THE MORE ADVENTUROUS:

The `datatool` package provides some hooks to allow you to make minor modifications to the default layout of `\DTLdisplaydb` and `\DTLdisplaylongdb`. These include the commands:

`\dtldisplaystarttab`

[Definition](#)

which is done at the start,

`\dtldisplayendtab`

[Definition](#)

which is done at the end and

`\dtldisplayafterhead`

[Definition](#)

which is done after the header row. In the case of `\DTLdisplaylongdb`, the hooks `\dtldisplaystarttab` and `\dtldisplayafterhead` are used before

2.6 Displaying Tabulated Data

and after the header row on each page of the `longtable`, but `\dtldisplayendtab` is only used on the *last* page of the `longtable`. You will need to use the `foot` option if you want to specify code that should appear at the bottom of every page of the `longtable`.

These three commands default to nothing, but you can redefine them before you display the data. For example, recall from [Volume 1](#) [93, §4.6.3] that the `booktabs` package [24] provides:

`\toprule[<wd>]`

Definition

for a top horizontal rule,

`\bottomrule[<wd>]`

Definition

for a bottom horizontal rule, and

`\midrule[<wd>]`

Definition

for a horizontal rule between rows. See if you can edit your example document to include these rules in the table displaying the list of products.

You can [download](#) or [view](#) the solution to this exercise.

2.7 Iteration

This section covers iterating over data. The `datatool` package provides a way of iterating over rows of a database, discussed in §2.7.1, but you may also find you need to iterate over a [comma-separated list](#), so this is discussed in §2.7.2. If you want to manually build up a list and then iterate over it, you may prefer to use `etoolbox`'s internal lists, discussed in §2.7.3. For a more general purpose loop ability, you may prefer to use `TEX`'s `\loop`, discussed in §2.7.4.

[FAQ: Repeating a command *n* times]

[FAQ: Repeating something for each 'thing' in a set]

2.7.1 Iterating Through a Database

The `datatool` package provides ways of iterating through a database and performing a task on each row of data. The two main commands are:

`\DTLforeach[<condition>]{<db-name>}{<assign>}{<body>}`

[Definition](#)

and its starred version:

`\DTLforeach*[<condition>]{<db-name>}{<assign>}{<body>}`

[Definition](#)

The unstarred version allows you to modify the data *stored internally* (that is, in `TEX`'s registers used by `datatool`, not in the original loaded or imported

2.7 Iteration

source). As it's more efficient to do any modifications in your spreadsheet or via SQL these datatool commands aren't covered here. Instead, all the examples in this document will use the read-only starred version, which compiles faster. The parameters for both versions are as follows:

`<db-name>` The label identifying the internal database.

`<assign>` A [comma-separated list](#) of `<cs>=<col-label>` assignments where `<cs>` is a control sequence that can be used as a placeholder in `<body>` and `<col-label>` is the label identifying the required column. *Spaces aren't ignored in this list* (except after `<cs>` as per TeX's normal behaviour). There is no check for the existence of `<cs>` so be careful you don't accidentally overwrite an existing command. You only need to assign control sequences to the columns whose values you intend to use in `<body>`. Assignments are performed with the [`\global`](#) prefix to ensure that [`\DTLforeach`](#) works correctly within a [`tabular`](#) (or similar) environment.

`<body>` The code to do for each row of data where the condition given in the optional argument is true.

`<condition>` This optional argument should be a conditional that follows

2.7 Iteration

the same syntax as the `\ifthenelse` command defined in the `ifthen` package [10]. For example, you can use:

`\equal{\langle text 1 \rangle}{\langle text 2 \rangle}`

Definition

to test if $\langle text 1 \rangle$ is equal to $\langle text 2 \rangle$.

The $\langle body \rangle$ is only applied to those rows where the condition is met. The default is `\boolean{true}`. If you're importing data from a `SQL` database, then it's better to apply any filtering in the `SELECT` statement.

You can prematurely terminate the list at the end of the current iteration by placing

`\dtlbbreak`

Definition

anywhere within $\langle body \rangle$.

For example, to simply print each surname in the people data:

`\DTLforeach*{\people}{\Surname=surname}{\Surname. }`

Input

Using the sample `people.csv` file, this produces:

2.7 Iteration

Parrot. Canary. Zebra. Arara. Duck. Canary.

Output

To just print the forenames of the people whose surname is “Canary”:

```
\DTLforeach*  
  [\equal{\Surname}{Canary}]\% condition  
  {people}\% database  
  {\Surname=surname,\Forenames=forenames}\% assignments  
  {\Forenames.\ }\% body
```

↑ Input

↓ Input

which produces:

Mabel. Fred.

Output

EXAMPLE 6. ITERATING THROUGH DATA

Recall from [Example 5](#) that data imported from an Excel .xls file doesn’t include any of the formatting used by the spreadsheet, so [Table 2.3](#) (produced using `\DTLdisplaydb`) didn’t display the numerical data as currency. Instead of using `\DTLdisplaydb` we can use `\DTLforeach*` to display the table and use `\dtlround` (described in [§2.1.3](#)) to round the values to two decimal places.

2.7 Iteration

↑ Input

```
\begin{table}
\caption{Formatted data imported from \texttt{shop.xls}}
\label{tab:xlsproducts2}
\centering
\begin{tabular}{lrr}
\multicolumn{1}{c}{\bfseries Product} &
\multicolumn{1}{c}{\bfseries Price (ex VAT)} &
\multicolumn{1}{c}{\bfseries Price (inc VAT)}%
\DTLforeach*{xlsproducts}{%
{%
\Product=Product,%
\exPrice=Price (ex VAT),%
\incPrice=Price (inc VAT)%
}%
{%
\\ \Product &
\dtlround{\exPrice}{\exPrice}{2}\pounds\exPrice &
\dtlround{\incPrice}{\incPrice}{2}\pounds\incPrice
}%
\end{tabular}
```

2.7 Iteration

```
\end{table}
```

↓ Input

which produces **Table 2.4**. Note that the new row command `\\"` is put at the start of `<body>` to ensure a new line starts after the header entries. It's usually best to put `\\"` at the start of `<body>` as it may cause a problem if it's placed later in that argument. You can [download](#) or [view](#) the complete document.

Output

Table 2.4 Formatted Data Imported From `shop.xls`

Product	Price (ex VAT)	Price (inc VAT)
Ink cartridge	£25.00	£30.00
Mouse mat	£12.00	£14.40
USB stick	£15.00	£18.00
Pen	£2.50	£3.00

2.7 Iteration

Table 2.5 Hardback Books

Id	Author	Title
5	Sir Quackalot	The Return of Duck and Goose
6	Sir Quackalot	The Adventures of Duck and Goose
10	Bor Ing	'Duck and Goose': an allegory for modern times?

Table 2.6 Paperback Books

Id	Author	Title
1	Sir Quackalot	The Adventures of Duck and Goose
2	Sir Quackalot	The Return of Duck and Goose
3	Sir Quackalot	More Fun with Duck and Goose
4	Sir Quackalot	Duck and Goose on Holiday
7	A. Parrot	My Friend is a Duck

Table 2.7 Ebooks

Id	Author	Title
8	Prof Macaw	Annotated Notes on the 'Duck and Goose' chronicles
9	Polly Parrot	'Duck and Goose' Cheat Sheet for Students

EXERCISE 2. ITERATING THROUGH DATA

Create a document that loads the sample `people.csv` file (or the `people` SQL table) and displays the three tables shown in Tables 2.5, 2.6 and 2.7 using `\DTLforeach*`. You can [download](#) or [view](#) the solution to this exercise.

2.7.2 Iterating Over a Comma-Separated List

The `etoolbox` package [51] provides:

`\docsclist{<item1,item2,...>}`

Definition

This iterates over the given `comma-separated list` and does

`\do{<item>}`

Definition

at each iteration, where `<item>` is the current item in the list. It's up to the user to define `\do` before using `\docsclist`. For example:

```
\renewcommand{\do}[1]{#1. }%
\docsclist{Parrot,Canary,Zebra,Arara,Duck}
```

↑ Input

↓ Input

2.7 Iteration

produces:

Parrot. Canary. Zebra. Arara. Duck.

Output

Alternatively you can provide your own handler instead of `\do` using

`\forcsvlist{<handler cs>}{{<list>}}`

Definition

For example:

```
\newcommand{\mylistitem}[1]{#1. }%
\forcsvlist{\mylistitem}{Parrot,Canary,Zebra,Arara,Duck}
```

↑ Input

↓ Input

which again produces:

Parrot. Canary. Zebra. Arara. Duck.

Output

 The argument `\docslist` (and `\forcsvlist`) doesn't get expanded, so if you try:

↑ Input

2.7 Iteration

```
\newcommand*\mylist{Parrot,Canary,Zebra,Arara,Duck}%
\renewcommand{\do}[1]{#1. }%
\docsclist{\mylist}
```



then you'll only have a list with a single item (`\mylist`) so you'll just have the one iteration

```
\do{\mylist}
```

Input

which just produces:

Parrot,Canary,Zebra,Arara,Duck.

Output

Instead you need to make sure the argument is expanded before it's processed by `\docsclist`:

```
\newcommand*\mylist{Parrot,Canary,Zebra,Arara,Duck}%
\renewcommand{\do}[1]{#1. }%
\expandafter\docsclist\expandafter{\mylist}
```



2.7 Iteration

The `\expandafter` commands may look a bit confusing but the syntax is

`\expandafter <token 1><token 2>`

Definition

This makes TeX expand `<token 2>` before it processes `<token 1>`, so it's equivalent to:

`<token 1>(expansion of token 2)`

Therefore

`\expandafter \docslist \expandafter`
 ↑ ↑
 `<token 1>` `<token 2>`

means that TeX must expand the thing after `\docslist` before it does `\docslist` (step ① in Figure 2.9), but that thing happens to be another `\expandafter`:

`\expandafter {\mylist`
 ↑ ↖
 `<token 1>` `<token 2>`

This means that before TeX processes the left brace character `{` it must first expand the `token` after it (step ②), so that `\mylist` is replaced with its definition (step ③).

2.7 Iteration

So TeX starts out with the first `\expandafter`, skips over `\docslist` ① and does the second `\expandafter` which makes TeX skip over the open brace ② and expand `\mylist`, which replaces `\mylist` with its definition ③.

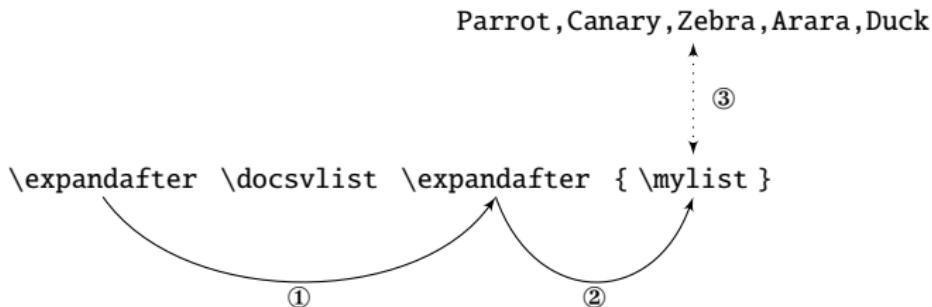


Figure 2.9 Processing `\expandafter`

Once `\mylist` has been expanded, TeX then goes back to the `\docslist` command, which is now in the form:

`\docslist{Parrot, Canary, Zebra, Arara, Duck}`

In the case of `\forcsvlist`, this becomes more complicated as the list is the second argument. However, since the first argument is just a sin-

2.7 Iteration

gle control sequence it doesn't need to be grouped and can therefore be skipped over with another `\expandafter`. For example:

```
\newcommand*\mylist{Parrot,Canary,Zebra,Arara,Duck}%
\newcommand{\mylistitem}[1]{#1. }%
\expandafter\forcsvlist\expandafter\mylistitem\expandafter{\mylist}
```

↑ Input

↓ Input

The `datatool` package also provides some [comma-separated list](#) related commands:

`\DTLifinlist{<item>}{<list>}{{<true>}}{{<false>}}`

Definition

This checks if the given item is in `<list>`. A one-level expansion is performed on `<list>` but not on `<item>`.

`\DTLnumitemsinlist{<list>}{{<cs>}}`

Definition

This counts the number of non-empty items in `<list>` and stores the result in the control sequence `<cs>`. Again, a one-level expansion is performed on `<list>`.

2.7 Iteration

EXAMPLE:

```
\newcommand{\mylist}{Parrot,Canary,Zebra,Arara,Duck}%
Parrot
\DTLifinlist{Parrot}{\mylist}{is}{isn't}
in the list.
Number of items in the list:
\DTLnumitemsinlist{\mylist}{\numitems}\numitems.
```

↑ Input

↓ Input

produces:

Parrot is in the list. Number of items in the list: 5.

Output

SPACES

Remember what I mentioned on page 22 about being careful of spaces? Here's an illustration of unexpected behaviour involving spaces in lists:

```
\renewcommand{\do}[1]{``#1'' . }%
\docs{ Parrot , Canary , Zebra , Arara , Duck }
```

↑ Input

↓ Input

2.7 Iteration

This produces:

“Parrot ”. “Canary ”. “Zebra ”. “Arara ”. “Duck ”.

Output

The leading spaces have been ignored but the trailing spaces are still present.

The \LaTeX kernel has a command that can also iterate through a **comma-separated list** but it's an **internal command**:

`\@for<cs>:=<list>\do{<body>}`

Definition

This iterates through $\langle list \rangle$ and assigns the control sequence $\langle cs \rangle$ to the current item in the list so that it can be used as a placeholder in $\langle body \rangle$. Note that in this context $\text{\do{}}$ doesn't refer to etoolbox's $\text{\do{}}$ handler macro but is used as an argument delimiter (so it's like plain \TeX syntax rather than \LaTeX syntax). Since \@for is an **internal command** it should only be used in a package or class file. If it has to be used in the document, it should be placed between \makeatletter and \makeatother like this:

```
\makeatletter
\@for\thisitem:=Parrot,Canary,Zebra,Arara,Duck\do{\thisitem. }
\makeatother
```

↑ Input

↓ Input

2.7 Iteration

This produces:

Parrot. Canary. Zebra. Arara. Duck.

Output

The `\@for` command expands the list so:

```
\newcommand*\mylist{Parrot,Canary,Zebra,Arara,Duck}%
\makeatletter
\@for\thisitem:=\mylist\do{\thisitem. }
\makeatother
```

↑ Input

↓ Input

produces the same result as above. However, now let's look at what happens when we introduce spaces into the list again:

```
\makeatletter
\@for\thisitem:= Parrot , Canary , Zebra , Arara , Duck \do
{``\thisitem''. }
\makeatother
```

↑ Input

↓ Input

2.7 Iteration

This produces:

“ Parrot ”. “ Canary ”. “ Zebra ”. “ Arara ”. “ Duck ”.

Output

In this case both the leading and trailing spaces have been retained.

The etextools package [15] also provides commands to iterate over lists. For example:

\csvloop[⟨auxiliary commands⟩]{⟨list⟩}

Definition

The \csvloop macro iterates over the comma-separated list given in ⟨list⟩ (which may be a macro that expands to a list) and at each iteration does ⟨auxiliary commands⟩. If this optional argument is missing, \do is assumed, which behaves in the same way as with etoolbox’s \docsvlist command. So

```
\renewcommand{\do}[1]{#1. }%
\csvloop{Parrot,Canary,Zebra,Arara,Duck}
```

↑ Input

↓ Input

produces

2.7 Iteration

Parrot. Canary. Zebra. Arara. Duck.

Output

Now let's try with spaces again:

```
\renewcommand{\do}[1]{``#1''. }%
\csvloop{ Parrot , Canary , Zebra , Arara , Duck }
```

↑ Input

↓ Input

This produces:

" Parrot ". "Canary ". "Zebra ". "Arara ". "Duck ".

Output

In this case the leading space has been retained on the first item, but not on any of the other items, but all the trailing spaces have been retained.

⚠ This section introduces one of etextools commands for illustrative purposes, but some of the commands in etextools and etoolbox conflict. For example, both packages define a command called \forlistloop but the syntax is incompatible. Since datatool automatically loads etoolbox this means that datatool and etextools may also conflict.

Another package is pgffor (part of the pgf bundle [102]) which provides:

2.7 Iteration

```
\foreach <variables> [<options>] in {<list>}{{<body>}}
```

Definition

where *<list>* is either an explicit **comma-separated list** or a control sequence that expands to a comma-separated list. The syntax can get quite complicated, but the simplest version is in the form:

```
\foreach \thisitem in {Parrot,Canary,Zebra,Arara,Duck}  
{\thisitem. }
```

↑ Input

↓ Input

which produces:

Parrot. Canary. Zebra. Arara. Duck.

Output

Now let's try with spaces:

```
\foreach \thisitem in { Parrot , Canary , Zebra , Arara , Duck }  
{\``\thisitem''. }
```

↑ Input

↓ Input

2.7 Iteration

This produces:

“Parrot ”. “Canary ”. “Zebra ”. “Arara ”. “Duck ”.

Output

which is the same result as with `\docslist` where the trailing spaces have been retained but not the leading spaces.

This is why it’s important to ensure any spurious spaces are removed from `comma-separated lists` in your source code. It may be that some commands trim all spaces (for example in the optional argument of `\usepackage` or `\documentclass`),⁶ while some commands only trim leading spaces (as with `\docslist` and `\foreach`) but others don’t trim any spaces (such as `\cite`, which internally uses `@for`). Remember that you can comment out space caused by the `EOL` character, and spaces at the start of lines are automatically discarded by TeX, so to make your code clearer you can do, for example:

```
|  
↑ Input  
\docslist  
{%  
 Parrot,%
```

⁶In fact, *all* spaces are stripped from the optional argument of those two commands, so it’s technically possible to do, say, `\usepackage[dr aft]{graphics}` although I don’t recommend you do this.

2.7 Iteration

```
Canary,%  
Zebra,%  
Arara,%  
Duck%  
}
```

↓ Input

If you comment an **EOL** character that would naturally be discarded (for example, following a control sequence) there's no harm done, but if you forget to comment an unwanted **EOL** character, you can end up with weird spaces in your document or an error message.

EMPTY ITEMS

In addition to watching out for spurious spaces, you also need to consider what happens if you have an empty item in your list. Do you want empty items skipped or should they be processed either in the same way as the other items or by displaying a missing data symbol, such as an em-dash? Let's look at how the above **comma-separated list** processing commands deal with this type of situation:

```
\renewcommand{\do}[1]{``#1''} . }
```

↑ Input

2.7 Iteration

```
\docs{list}{Parrot,Canary,Zebra,,Arara,Duck,}

\csvloop{,Parrot,Canary,Zebra,,Arara,Duck,}

\foreach\thisitem in {,Parrot,Canary,Zebra,,Arara,Duck,}
{``\thisitem''. }

\makeatletter
\@for\thisitem:=,Parrot,Canary,Zebra,,Arara,Duck,\do
{``\thisitem''. }
\makeatother
```

↓ Input

The result is:

```
"Parrot". "Canary". "Zebra". "Arara". "Duck".
"Parrot". "Canary". "Zebra". "Arara". "Duck".
"". "Parrot". "Canary". "Zebra". "". "Arara". "Duck". "".
"". "Parrot". "Canary". "Zebra". "". "Arara". "Duck". "".
```

↑ Output

2.7 Iteration

So `\docslist` and `\csvloop` both skip empty items but `\foreach` and `\@for` don't.

EXERCISE 3. ITERATING THROUGH A LIST

Create a document that defines the commands:

```
\newcommand*{\mylistI}{A,B,C,D}
\newcommand*{\mylistII}{a,b,c,d}
```

↑ Input

↓ Input

then create the `tabular` environment shown in Table 2.8 using `\docslist` to construct columns 2 to 5 for each row.

Table 2.8 Lists for Iteration Exercise

List 1: A B C D

List 2: a b c d

Next, redefine `\mylistII` so the third item is empty and redo the `tabular` environment. Finally, define a command called `\missingdata` that does nothing and redefine `\mylistII` so the third item is `\missingdata` and redo the `tabular` environment.

You can [download](#) or [view](#) the solution to this exercise.

2.7.3 Iteration With etoolbox's Internal Lists

The [previous section](#) looked at iterating over a [comma-separated list](#), but it may be that you need to first construct a list before iterating over it. This can be done efficiently via etoolbox's internal lists. These don't use a comma as a separator so it's useful for lists where items may potentially contain commas. For example, suppose I want to make a list called, say, `\mylist`, then I first need to define an empty list:

```
\newcommand*{\mylist}{}
```

Input

An item can be added to the list using:

```
\listadd{\list cs}{\item}
```

Definition

where `\list cs` is the command used to store the list (`\mylist` in the above example) and `\item` is the item to add to the list. Note that the `\item` doesn't get expanded and a blank item won't be added to the list. For example:

2.7 Iteration

```
\listadd{\mylist}{Parrot}  
\listadd{\mylist}{Parrot, Jr}
```

↑ Input

↓ Input

If the item needs to be expanded before being added to the list, you can use:

```
\listadd{<list cs>}{<item>}
```

Definition

As with `\listadd`, a blank item won't be added to the list. For example:

```
\newcommand*{\Name}{Canary}  
\listadd{\mylist}{\Name}
```

↑ Input

↓ Input

There are also similar commands where you supply the name of the list macro without the leading backslash:

```
\listcsadd{<list csname>}{<item>}
```

Definition

(unexpanded item) and

2.7 Iteration

\listcseadd{\list csname}{\item}

Definition

(expanded item). For example:

```
\renewcommand*\Name{Zebra}
\listcseadd{mylist}{\Name}
\listcsadd{mylist}{Arara}
```

↑ Input

↓ Input

These commands all use local assignments, so they're limited to the current scope. There are analogous commands that use global assignments:

\listgadd{\list cs}{\item}

Definition

(global version of \listadd)

\listxadd{\list cs}{\item}

Definition

(global version of \listeadd)

\listcsgadd{\list csname}{\item}

Definition

(global version of \listcsadd) and

2.7 Iteration

`\listcsxadd{\list csname}{\item}`

Definition

(global version of `\listcseadd`).

You can test if an item is in a list using:

`\ifinlist{\item}{\list cs}{\true}{\false}`

Definition

(No expansion is performed on `\item`.) For example:

```
Parrot  
\ifinlist{Parrot}{\mylist}{is}{isn't}  
in the list.
```

↑ Input

↓ Input

If you want to test the expansion of an item, you can use:

`\xifinlist{\item}{\list cs}{\true}{\false}`

Definition

For example:

```
\newcommand*\Name{Parrot}%  
\Name\_\xifinlist{\Name}{\mylist}{is}{isn't} in the list.
```

↑ Input

↓ Input

2.7 Iteration

There are also analogous commands where the list control sequence name (without the leading backslash) is supplied:

`\ifinlistcs{<item>}{{<list csname>}}{<true>}{<false>}`

Definition

for the non-expanded version and

`\xifinlistcs{<item>}{{<list csname>}}{<true>}{<false>}`

Definition

for the item expansion version.

Once you've added all your items to the list, you can iterate over the list using:

`\dolistloop{{<list cs>}}`

Definition

where `<list cs>` is the control sequence storing the list (`\mylist` in the above examples). If you prefer to supply the control sequence name without the leading backslash, you can use:

`\dolistcsloop{{<list csname>}}`

Definition

Both these commands use `\do{<item>}` at each iteration, in the same way as for `\docsclist` described earlier. For example:

↑ Input

2.7 Iteration

```
\newcommand*{\mylist}{}
\listadd{\mylist}{Parrot}
\listadd{\mylist}{Canary}
\listadd{\mylist}{Zebra}
\listadd{\mylist}{Arara}
\listadd{\mylist}{Duck}
\renewcommand*{\do}[1]{#1. }
\do{\listloop{\mylist}}
```

↓ Input

produces:

Parrot. Canary. Zebra. Arara. Duck.

Output

Alternatively, you can provide your own handler instead of using `\do`:

```
\forlistloop{<handler cs>}{<list cs>}
```

Definition

where `<handler cs>` is the command to use on each iteration of the list and `<list cs>` is the list control sequence. If you prefer to supply the list control sequence name without the leading backslash you can use:

```
\forlistcsloop{<handler cs>}{{<list csname>}}
```

Definition

EXERCISE 4. INTERNAL LISTS

Create a document that loads the sample `booklist.csv` file or the `books` SQL table. Then create an internal list that contains a list of all the book titles, without repetition. For example, if a title has both a hardback and paperback edition only add that title once rather than twice.

To test the list, iterate through it and display each item of the list. (If you like, you can just use a paragraph break between items rather than using a `tabular` environment.) The result should look like:

```
The Adventures of Duck and Goose
The Return of Duck and Goose
More Fun with Duck and Goose
Duck and Goose on Holiday
My Friend is a Duck
Annotated Notes on the 'Duck and Goose' chronicles
'Duck and Goose' Cheat Sheet for Students
'Duck and Goose': an allegory for modern times?
```

↑ Output

↓ Output

You can [download](#) or [view](#) the solution to this exercise.

2.7 Iteration

2.7.4 General Iteration with T_EX's \loop

If you want a more general purpose way of repeating a block of code, you can use

\loop <code>\if... \repeat

Definition

This repeats <code> while the given condition is true. The \if... part should be one of T_EX's conditionals, such as \ifnum, without the terminating \fi.

EXAMPLE:

For example, to print the numbers from 1 to 10:

```
\newcount\mycount
\loop
  \advance\mycount by 1\relax
  \the\mycount.
\ifnum\mycount<10
\repeat
```

↑ Input

↓ Input

2.7 Iteration

This produces:

1. 2. 3. 4. 5. 6. 7. 8. 9. 10.

Output

2.7.5 Iteration Tips and Tricks

This section describes some advanced techniques that you may or may not need to know, so feel free to skip it.

Recall from §2.7.1 I had:

\DTLforeach*{people}{\Surname=surname}{\Surname. }

Input

which, when using the sample `people.csv` file, produced:

Parrot. Canary. Zebra. Arara. Duck. Canary.

Output

Suppose instead I wanted to produce:

Parrot; Canary; Zebra; Arara; Duck; Canary.

Output

If I try:

2.7 Iteration

```
\DTLforeach*{people}{\Surname=surname}{\Surname; }.
```

Input

I get:

Parrot; Canary; Zebra; Arara; Duck; Canary; .

Output

(there's an unwanted semi-colon and space before the terminating full stop)
and if I try:

```
\DTLforeach*{people}{\Surname=surname}{; \Surname}.
```

Input

I get:

; Parrot; Canary; Zebra; Arara; Duck; Canary.

Output

(The unwanted semi-colon and space are now at the start.)

Neither or these are quite right. Here's a way of achieving the desired output:

```
\newcommand{\surnamesep}{%
  \renewcommand{\surnamesep}{; }%
}
\DTLforeach*{people}{\Surname=surname}{\surnamesep\Surname}.
```

↑ Input

↓ Input

2.7 Iteration

This may look a bit weird at first sight, but here's how it works:

- On the first iteration

\surnamesep\Surname

Input

is equivalent to

\renewcommand{\surnamesep}{; } \Surname

Input

That is, \surnamesep is redefined to ; (semicolon followed by a space) without displaying anything and the first surname is printed.

- On the next iteration

\surnamesep\Surname

Input

is equivalent to just

; \Surname

Input

(because \surnamesep has just been redefined to ; so a semi-colon followed by a space followed by the second surname is printed.)

2.7 Iteration

Since `\surnamesep` doesn't get redefined any more, it remains the same for the rest of the loop.

⚠ Be careful if the contents of `<body>` are localised (for example, if it's in a `tabular` environment, as in [Example 6](#)) since `\renewcommand` only has a local effect. Instead you can use `\gdef` command described in [§2.1.1](#):

```
\newcommand{\surnamesep}{%
  \gdef\surnamesep{; }%
}
\DTLforeach*{people}{\Surname=surname}{\surnamesep\Surname}.
```

↑ Input

↓ Input

Alternatively you can do a global `\let` after you've redefined the command:

```
\newcommand{\surnamesep}{%
  \renewcommand\surnamesep{; }%
}
\global\let\surnamesep\surnamesep
```

↑ Input

2.7 Iteration

```
}%  
\DTLforeach*{people}{\Surname=surname}{\surnamesep\Surname}.
```

↓ Input

The datatool package defines the command:

```
\DTLiffirstrow{\langle true \rangle}{\langle false \rangle}
```

Definition

designed for use within the *body* of `\DTLforeach` so I could just do:

```
\DTLforeach*  
{people}% database  
{\Surname=surname}% assignment list  
{\DTLiffirstrow{}{; }\Surname}.
```

↑ Input

↓ Input

However, the technique described above can be used in more general situations. For example, suppose I want to use etoolbox's `\docsvlist`, described in §2.7.2, I could do:

2.7 Iteration

```
\newcommand{\surnamesep}{%
  \renewcommand{\surnamesep}{; }%
}%
\renewcommand\do[1]{\surnamesep#1}%
\docs{list}{Parrot,Canary,Zebra,Arara,Duck}.
```

↑ Input

which produces:

Parrot; Canary; Zebra; Arara; Duck.

Output

The `datatool` package also defines the command:

```
\DTLiflastrow{<true>}{<false>}
```

Definition

As with `\DTLiffirstrow`, this command is designed for use within the `<body>` of `\DTLforeach` (or `\DTLforeach*`). For example:

```
\DTLforeach*%
{people}%
{\Surname=surname}%
assignment list
```

↑ Input

2.7 Iteration

```
{%  
 \DTLiffirstrow{}{\DTLiflastrow{ and }{, }%  
 \Surname  
 }.
```

↓ Input

produces:

Parrot, Canary, Zebra, Arara, Duck and Canary.

Output

So how can we do the equivalent for a general **comma-separated list** rather than using **\DTLforeach**? One possible method is described in the example below.

EXAMPLE 7. LIST OF NAMES

This is slightly more complicated but it uses a similar technique to earlier:

```
\newcommand*{\surnamesep}{%}  
\newcommand*{\lastsurname}{%}  
\newcommand*{\prelastsurname}{%}  
\renewcommand*{\do}[1]{%  
 \surnamesep  
 \lastsurname
```

↑ Input

2.7 Iteration

```
\renewcommand{\lastsurname}{%
  \renewcommand{\surnamesep}{, }%
  \renewcommand{\prelastsurname}{ and }%
  #1%
}%
}%
\docs{Parrot,Canary,Zebra,Arara,Duck}%
\prelastsurname \lastsurname
```

↓ Input

This produces:

Parrot, Canary, Zebra, Arara and Duck

Output

Here's how it works:

1. On the first iteration (`\do{Parrot}`) `\surnamesep` and `\lastsurname` do nothing. Then `\lastsurname` is redefined via:

```
\renewcommand{\lastsurname}{%
  \renewcommand{\surnamesep}{, }%
  \renewcommand{\prelastsurname}{ and }%
  Parrot%
}
```

2.7 Iteration

(The `#1` has been replaced with the argument passed to `\do`.) So far nothing has been displayed.

If this was the only item in the list, the loop would end and then:

- `\prelastsurname` would be done, but this is currently nothing.
- `\lastsurname` would be done, which is now set to redefine a couple of commands that are no longer needed (`\surnamesep` and `\prelastsurname`) and then displays “Parrot”.

Therefore, if the list only has one element, it just displays that element. However, since the list has more than one element, we have nothing displayed and we move on to the next item in the list.

2. On the second iteration (`\do{Canary}`) `\surnamesep` still does nothing but `\lastsurname` now does:

```
\renewcommand{\surnamesep}{, }%
\renewcommand{\prelastsurname}{ and }%
Parrot%
```

2.7 Iteration

So `\surnamesep` gets redefined to do `,` (that is a comma followed by a space) and `\prelastsurname` gets redefined to do `and`. After these redefinitions, the word “Parrot” is then displayed. Next `\lastname` is redefined via:

```
\renewcommand{\lastname}{%
  \renewcommand{\surnamesep}{, }
  \renewcommand{\prelastsurname}{ and }
  Canary%
}
```

Therefore, by the end of the second iteration, the only text displayed is “Parrot”. If this happened to be the last item in the list, the loop would end and then:

- `\prelastsurname` would be done, which now displays “ and ”.
- `\lastname` would be done, which redefines some commands we no longer need (`\surnamesep` and `\prelastsurname`) and then displays “Canary”.

Therefore, if the list only contained `Parrot,Canary` then just the text “Parrot and Canary” would be displayed. However, as there are still more items left, the only text displayed so far is “Parrot”.

2.7 Iteration

3. On the third iteration (`\do{Zebra}`) `\surnamesep` now displays a comma followed by a space and `\lastsurname` does:

```
\renewcommand{\surnamesep}{, }%
\renewcommand{\prelastsurname}{ and }%
Canary%
```

Then `\lastsurname` is redefined via:

```
\renewcommand{\lastsurname}{%
\renewcommand{\surnamesep}{, }%
\renewcommand{\prelastsurname}{ and }%
Zebra%
}
```

So we have thus far produced the text “Parrot, Canary”. If this happened to be the last item in the list, the loop would end and then:

- `\prelastsurname` would display “ and ”
- `\lastsurname` would redefine some commands that we no longer need (`\surnamesep` and `\prelastsurname`) and then display “Zebra”.

2.7 Iteration

The remaining iterations follow the same pattern as this third iteration.

If you plan to use this method more than once, you might prefer to define a new command. For example:

↑ Input

```
% set up defaults so we don't get an error
% when we try to redefine these commands
\newcommand*{\surnamesep}{}
\newcommand*{\lastsurname}{}
\newcommand*{\prelastsurname}{}
% define the new command to process a list of names:
\newcommand*{\displaynames}[1]{%
    % initialise:
    \renewcommand*{\surnamesep}{}
    \renewcommand*{\lastsurname}{}
    \renewcommand*{\prelastsurname}{}
    % set up list handler:
    \renewcommand*{\do}[1]{%
        \surnamesep
        \lastsurname
    }
}
```

2.7 Iteration

```
\renewcommand{\lastsurname}{%
  \renewcommand{\surnamesep}{, }%
  \renewcommand{\prelastsurname}{ and }%
  ##1%
}%
}%
\docsvlist{#1}%
\prelastsurname \lastsurname
}
```

↓ Input

Since we have nested definitions of commands that take a parameter we need to be careful how we reference the parameter. In the above `#1` refers to the argument of `\displaynames` (the outer command) and `##1` refers to the argument of `\do` (the inner command).

In this case, it's better to define your own handler macro and use `\forlistloop` instead of `\docsvlist`:

↑ Input

```
% set up defaults so we don't get an error
% when we try to redefine these commands
\newcommand*\surnamesep{}%
```

2.7 Iteration

```
\newcommand*{\lastsurname}{}
\newcommand*{\prelastsurname}{}
% define the handler macro:
\newcommand*{\dodisplayname}[1]{%
    \surnamesep
    \lastsurname
    \renewcommand{\lastsurname}{%
        \renewcommand{\surnamesep}{, }
        \renewcommand{\prelastsurname}{ and }%
        #1%
    }%
}
% define the new command to process a list of names:
\newcommand*{\displaynames}[1]{%
    % initialise:
    \renewcommand*{\surnamesep}{}
    \renewcommand*{\lastsurname}{}
    \renewcommand*{\prelastsurname}{}
    % iterate through list:
    \forcsvlist{\dodisplayname}{#1}%
    % finish off:
    \prelastsurname \lastsurname
```

2.7 Iteration

```
}
```

↓ Input

This removes one of the nested redefinitions and the need to use `##1` instead of `#1`.

You can [download](#) or [view](#) a complete document.

EXERCISE 5. OXFORD COMMA

Some people always use the Oxford comma, some people never use it, and some people only use it where its lack would cause ambiguity. The purpose of this exercise is not to engage in a heated debate over whether or not it should be used. It's simply an exercise in iteration techniques. For those who've never heard of the Oxford comma, it's a comma that's placed after the penultimate item in a list of three or more items before the "and". For example: Parrot, Canary, and Zebra.

For this exercise, see if you can adapt the definition of `\displaynames` from the end of the previous example so that it uses the Oxford comma. To test it:

↑ Input

2.7 Iteration

```
\displaynames{Parrot,Canary,Zebra,Arara,Duck}  
  
\displaynames{Parrot,Canary,Zebra,Arara}  
  
\displaynames{Parrot,Canary,Zebra}  
  
\displaynames{Parrot,Canary}  
  
\displaynames{Parrot}
```

↓ Input

should produce:

```
Parrot, Canary, Zebra, Arara, and Duck  
Parrot, Canary, Zebra, and Arara  
Parrot, Canary, and Zebra  
Parrot and Canary  
Parrot
```

↑ Output

You can [download](#) or [view](#) the solution to this exercise.

↓ Output

2.8 Fetching Data From a Given Row

It may be that you don't want to iterate through the entire data but just want to fetch information from a particular row. The `datatool` package provides a number of ways to do this, but this book is just going to cover three commands:

`\DTLassign{<db-name>}{<row-idx>}{<assign list>}`

Definition

This applies the assignment list to the row given by `<row-idx>`. (Indices start from 1.)

`\DTLassignfirstmatch{<db-name>}{<col-label>}{<value>}{<assign list>}`

Definition

This applies the assignment list to the first row where the entry in the column identified by `<col-label>` exactly matches the given value. Note that no expansion is performed on `<value>`.

`\xDTLassignfirstmatch{<db-name>}{<col-label>}{<value>}{<assign list>}`

Definition

This applies the assignment list to the first row where the entry in the column identified by `<col-label>` exactly matches a *one-level expansion* of `<value>`.

In each case, `<db-name>` is the label identifying the data and `<assign list>` is the **comma-separated list** of assignments, as used by `\DTLforeach`

2.8 Fetching Data From a Given Row

and `\DTLforeach*`.

EXAMPLE 8. FETCHING THE DATA FROM ROW 1

Suppose I just want information from the first row of data in my sample `people.csv` file. Then I can use `\DTLassign`, like this:

```
\DTLassign{people}{1}{%
  \Surname=surname,%
  \Title=title,%
  \AddressI=address1,%
  \AddressII=address2,%
  \Town=town,%
  \County=county,%
  \Postcode=postcode%
}
```

↑ Input

↓ Input

Remember to make sure you comment out the unwanted `EOL` characters, as shown above, or you'll get an error caused by spurious spaces (recall the note about spaces on page 162). Now that the data has been fetched, it can be used. For example, to just display the details in a `tabular` environment:

2.8 Fetching Data From a Given Row

```
\begin{tabular}{l}
>Title\_\_Surname\\
>AddressI\\
>AddressII\\
>Town\\
>County\\
>Postcode
\end{tabular}
```

↑ Input

↓ Input

This produces:

Miss Parrot
42 The Lane

↑ Output

Some Town
Noshire
AB1 2XY

↓ Output

2.8 Fetching Data From a Given Row

(You can [download](#) or [view](#) a complete document.)

Note that there is a blank entry caused by missing data in the address2 column. If this example was changed to use the `people` SQL table instead, the result would appear as:

```
Miss Parrot
42 The Lane
NULL
Some Town
Noshire
AB1 2XY
```

↑ Output

↓ Output

See [§2.9](#) on how to deal with null or empty entries. (You can also [download](#) or [view](#) a complete document for the `SQL` version.)

Remember that if you're importing your data from a `SQL` database, there's no need to import all the data from the table if you don't require parts of it. Instead you can filter out all the unwanted rows in your `SELECT` statement. For example, if you wanted to fetch the data for just the customer whose surname is "Parrot", you can do:

2.8 Fetching Data From a Given Row

```
datatooltk --output customer.dbtex --sqldb samples --sqluser  
sampleuser --sql "SELECT * FROM people WHERE surname='Parrot'"
```

Shell

If you're not using **SQL** then you can fetch the relevant row using the afore mentioned **\DTLassignfirstmatch**, but it's less efficient.

EXAMPLE 9. FETCHING A CUSTOMER'S DETAILS

Suppose you only want the details from the customer whose surname matches "Parrot" in the sample **people.csv** file. This can be fetched using:

```
\DTLassignfirstmatch{people}{surname}{Parrot}{%  
  \Surname=surname,%  
  \Title=title,%  
  \AddressI=address1,%  
  \AddressII=address2,%  
  \Town=town,%  
  \County=county,%  
  \Postcode=postcode%  
}
```

↑ Input

↓ Input

2.8 Fetching Data From a Given Row

Now the details have been fetched, it can be used as in the previous example:

```
\begin{tabular}{l}
>Title\_\_Surname\\
\AddressI\\
\AddressII\\
\Town\\
\County\\
\Postcode
\end{tabular}
```

↑ Input

↓ Input

The result is the same as for the previous example. (You can [download](#) or [view](#) a complete document.)

Remember that `\DTLassignfirstmatch` performs an exact match without expansion. This means that if you do something like:

2.8 Fetching Data From a Given Row

↑ Input

```
\newcommand{\Name}{Parrot}
\DTLassignfirstmatch{people}{surname}{\Name}%
{%
  \Surname=surname,%
  \Title=title,%
  \AddressI=address1,%
  \AddressII=address2,%
  \Town=town,%
  \County=county,%
  \Postcode=postcode%
}
```



↓ Input

Then you'll get an error that no match was found. This is because you're effectively asking T_EX to find an entry that contains "\Name", but that control sequence doesn't appear in any of the entries, so there's no match. Instead, you need to use `\xDTLassignfirstmatch` which will internally replace \Name with its definition ("Parrot").

↑ Input

2.8 Fetching Data From a Given Row

```
\newcommand{\Name}{Parrot}
\xDTLassignfirstmatch{people}{surname}{\Name}%
{%
  \Surname=surname,%
  \Title=title,%
  \AddressI=address1,%
  \AddressII=address2,%
  \Town=town,%
  \County=county,%
  \Postcode=postcode%
}
```



↓ Input

EXAMPLE 10. FETCHING A CUSTOMER'S DETAILS (WITH EXPANSION)

In Example 8, I didn't access the country from the data. Let's modify that example so that it fetches the complete address for "Polly Parrot":

```
\DTLassignfirstmatch{people}{surname}{Parrot}%
  \Surname=surname,%
  \Title=title,%
  \AddressI=address1,%
```

↑ Input

2.8 Fetching Data From a Given Row

```
\Town=town,%
\County=county,%
\Postcode=postcode,%
\CountryCode=country%
}
```

↓ Input

Now let's try displaying the information:

```
\begin{tabular}{l}
>Title\_\_Surname\\
Address\\
Town\\
County\\
Postcode\\
CountryCode
\end{tabular}
```

↑ Input

↓ Input

This produces:

2.8 Fetching Data From a Given Row

```
Miss Parrot  
42 The Lane  
Some Town  
Noshire  
AB1 2XY  
gb
```

↑ Output

If this is intended for, say, a letter to a customer, then the country code really needs to be converted to the country's name. That information is stored in the sample `country-codes.csv` file, so that also needs to be loaded. Therefore the document should have:

```
\DTLloaddb{people}{people.csv}  
\DTLloaddb{countries}{country-codes.csv}
```

↑ Input

↓ Input

Once the `\CountryCode` has been assigned via `\DTLassign`, as shown above, the code can be converted to a name:

2.8 Fetching Data From a Given Row

```
\xDTLassignfirstmatch{countries}{code}%
{\CountryCode}{\CountryName=name}
```

↑ Input

↓ Input

Now the address can be displayed including the country name:

```
Miss Parrot
42 The Lane
Some Town
Noshire
AB1 2XY
United Kingdom
```

↑ Output

↓ Output

You can [download](#) or [view](#) this example.

Remember that if you're using [SQL](#), it's much simpler to combine and filter using the `SELECT` statement:

2.8 Fetching Data From a Given Row

```
datatooltk --output customer.dbtex --sqldb samples  
--sqluser sampleuser --sql "SELECT title, surname, address1,  
address2, town, county, countries.name AS country, postcode  
FROM people, countries WHERE surname='Parrot' AND  
people.country = countries.code"
```

Shell

This creates a datatool (.dbtex) file called `customer.dbtex` that only contains the one row of data. The country name is now stored in the column labelled `country`. So you can just do:

```
\DTLloaddbtex{customer}{customer.dbtex}  
\DTLassign{customer}{1}{%  
  \Surname=surname,%  
  \Title=title,%  
  \AddressI=address1,%  
  \Town=town,%  
  \County=county,%  
  \Postcode=postcode,%  
  \CountryName=country%  
}
```

↑ Input

2.8 Fetching Data From a Given Row

```
\begin{tabular}{l}
>Title\_\_Surname\\
AddressI\\
Town\\
County\\
Postcode\\
CountyName
\end{tabular}
```

↓ Input

This produces the sample result as above. You can [download](#) or [view](#) this [SQL](#) version.

EXERCISE 6. FETCHING A ROW OF DATA

Modify the code from [Example 10](#) so that it fetches the address for Fred Canary from sample [people.csv](#) file.

You can [download](#) or [view](#) the solution to this exercise.

2.9 Null and Boolean Values

Recall from [Example 8](#) that there is a difference between the sample `people.csv` file and the corresponding `people` SQL table. Some of the entries in the `SQL` table have null values in the `address2` column whereas in the CSV file the values are empty rather than null. You can test for a null value using:

`\DTLifnull{\langle cs\rangle}{\langle true\rangle}{\langle false\rangle}`

[Definition](#)

where $\langle cs \rangle$ is a control sequence, $\langle true \rangle$ is what to do if $\langle cs \rangle$ is null and $\langle false \rangle$ is what to do if $\langle cs \rangle$ isn't null. To test for an empty value, you can use:

`\ifdefempty{\langle cs\rangle}{\langle true\rangle}{\langle false\rangle}`

[Definition](#)

which is provided by the `etoolbox` package (automatically loaded by `datatool`). The `datatool` package provides the command:

`\DTLifnullorempty{\langle cs\rangle}{\langle true\rangle}{\langle false\rangle}`

[Definition](#)

which is just a short cut for:

`\ifdefempty{\langle cs\rangle}{\langle true\rangle}{\DTLifnull{\langle cs\rangle}{\langle true\rangle}{\langle false\rangle}}`

[Input](#)

2.9 Null and Boolean Values

The other difference is the way boolean values have been stored. Both the `people.csv` file and the `SQL` `people` table used “1” to indicate a true value and “0” to indicate a false value in the `subscribed` field, but when the data was fetched from the SQL table, these values were converted to “true” and “false” in the `datatool (.dbtex)` file. There are a number of ways of testing whether a control sequence is equal to “true” or “1”. For example, the `etoolbox` package defines:

```
\ifdefstring{\cs}{\string}{\truepart}{\falsepart}
```

Definition

This tests if the control sequence `\cs` is defined to be `\string`. If it is, `\truepart` is done, otherwise `\falsepart` is done. For example:

```
\newcommand*\Subscribed{true}%
\ifdefstring{\Subscribed}{true}{Yes}{No}
```

↑ Input

↓ Input

produces:

Yes

Output

Similarly

2.9 Null and Boolean Values

```
\newcommand*\Subscribed{1}%
\ifdefstring{\Subscribed}{1}{Yes}{No}
```

↑ Input

↓ Input

produces:

Yes

Output

If you want to test for “true” or “1”, you can combine these:

```
\ifdefstring
{\Subscribed}{true}{ test
{Yes}}% condition true
{\ifdefstring{\Subscribed}{1}{Yes}{No}}
```

↑ Input

↓ Input

Alternatively you can use another etoolbox command:

```
\ifboolexpr{(expression)}{(true part)}{(false part)}
```

Definition

which evaluates *(expression)* and does *(true part)* if true, otherwise it does *(false part)*. In this case I’m going to use the **test** syntax:

2.9 Null and Boolean Values

```
\ifboolexpr
{
  test{\ifdefstring{\Subscribed}{true}} or
  test{\ifdefstring{\Subscribed}{1}}
}
{Yes}{No}
```

↑ Input

(For further details of the syntax used in `<expression>`, see the etoolbox documentation [51].) You might find it easier to define a command to do this. For example:

```
\newcommand{\ifcsbool}[3]{%
\ifboolexpr
{
  test{\ifdefstring{#1}{true}} or
  test{\ifdefstring{#1}{1}}
}
{#2}{#3}%
}
```

↑ Input

↓ Input

2.9 Null and Boolean Values

This has the syntax:

```
\ifcsbool{<cs>}{{true}}{{false}}
```

where $\langle cs \rangle$ is a control sequence. For example:

```
\ifcsbool{\Subscribed}{Yes}{No}
```

Input

or (recall from [Volume 1](#) [93, §8.2] the `\ding` command provided by `pi-font` [84]):

```
\ifcsbool{\Subscribed}{\ding{52}}{\ding{56}}
```

Input

EXAMPLE 44. DISPLAY CUSTOMER LIST (NULL VALUES)

To illustrate the difference between null and empty values, let's first look at what happens if we load the sample `people.csv` file and display the data using `\DTLdisplaydb`:

```
\DTLloaddb{people}{people.csv}

\begin{table}
  \caption{Customers (CSV)}
  \label{tab:peoplecsv}
```

↑ Input

2.9 Null and Boolean Values

```
\centering
\DTLdisplaydb{people}
[id,forenames,title,country,postcode,gender,dob]%
  omit these columns
{people}
\end{table}
```

↓ Input

This produces [Table 2.9](#). Now let's look at what happens if we fetch the `people` SQL table using:

```
datatooltk --output people.dbtex --sqluser sampleuser
--sqldb samples --sql "SELECT * FROM people"
```

Shell

and load the resulting `people.dbtex` file:

```
\DTLloaddbtex{\people}{people.dbtex}
```

Input

Now display the data:

```
\begin{table}
\caption{Customers (SQL)}
\label{tab:peoplesql}
```

↑ Input

2.9 Null and Boolean Values

```
\centering
\DTLdisplaydb
[id,forenames,title,country,postcode,gender,dob]%
  omit these columns
{\people}
\end{table}
```

↓ Input

This produces [Table 2.10](#).

You can use `\DTLifnull` or `\ifdefempty` to check for missing entries in the [SQL](#) or [CSV](#) data, respectively. Alternatively, use `\DTLifnullorempty` to adapt for either case. However, we now can't simply use `\DTLdisplaydb` but we can use `\DTLforeach*` to typeset the table instead:

```
\begin{table}
\caption{Customers (Check for Null and Boolean)}
\label{tab:peoplenullcheck}
\centering
\begin{tabular}{llllllc}
\multicolumn{1}{c}{\bfseries Surname} &
\multicolumn{1}{c}{\bfseries Address 1} &
\multicolumn{1}{c}{\bfseries Address 2} &
```

↑ Input

2.9 Null and Boolean Values

```
\multicolumn{1}{c}{\bfseries Town} &
\multicolumn{1}{c}{\bfseries County} &
\multicolumn{1}{c}{\bfseries Subscribed}%
\DTLforeach*{people}{\Surname=surname,\AddressI=address1,%
\AddressII=address2,\Town=town,\County=county,\Subscribed=subscribed}%
{%
  \\\Surname & \AddressI &
  \DTLifnullorempty{\AddressII}{\multicolumn{1}{c}{---}}{\AddressII}%
  & \Town &
  \DTLifnullorempty{\County}{\multicolumn{1}{c}{---}}{\County} &
  \ifcsbool{\Subscribed}{\ding{52}}{\ding{56}}%
}
\end{tabular}
\end{table}
```

↓ Input

(Recall `\ifcsbool` from [above](#).) This produces [Table 2.11](#).

You can [download](#) or [view](#) a complete document.

2.9 Null and Boolean Values

Table 2.9 Customers (CSV)

surname	address1	address2	town	county	subscribed
Parrot	42 The Lane		Some Town	Noshire	1
Canary	24 The Street	Some Village	Some Town	Noshire	0
Zebra	856 The Avenue		Some City	CA	1
Arara	Nenhuma Rua		São Paulo		1
Duck	1 The Street	Another Village	Some City	Imagineshire	0
Canary	24 The Street	Some Village	Some Town	Noshire	1

Table 2.10 Customers (SQL)

surname	address1	address2	town	county	subscribed
Parrot	42 The Lane	NULL	Some Town	Noshire	true
Canary	24 The Street	Some Village	Some Town	Noshire	false
Zebra	856 The Avenue	NULL	Some City	CA	true
Arara	Nenhuma Rua	NULL	São Paulo	NULL	true
Duck	1 The Street	Another Village	Some City	Imagineshire	false
Canary	24 The Street	Some Village	Some Town	Noshire	true

2.9 Null and Boolean Values

Table 2.11 Customers (Check for Null or Empty and Boolean)

Surname	Address 1	Address 2	Town	County	Subscribed
Parrot	42 The Lane	—	Some Town	Noshire	✓
Canary	24 The Street	Some Village	Some Town	Noshire	✗
Zebra	856 The Avenue	—	Some City	CA	✓
Arara	Nenhuma Rua	—	São Paulo	—	✓
Duck	1 The Street	Another Village	Some City	Imagineshire	✗
Canary	24 The Street	Some Village	Some Town	Noshire	✓

3. CORRESPONDENCE

This chapter covers writing letters (including mail merging), envelope labels and faxes. There are a number of classes you can use for this purpose. See, for example, the TeX Catalogue Topic Index [25] or the [letter topic](#). (If you're writing in German, you may want to consider [dinbrief](#) [8].)

[FAQ: Letters and the like]

For brevity, this book will only look at four such classes:

- letter** This is one of the base L^AT_EX classes that comes with all T_EX distributions. It's not very flexible, but it's simple to use and may suit your purposes if you're happy with the default layout. Since it's one of the base classes, it's stable and shouldn't conflict with common packages.
- scrlttr2** This is one of the KOMA-Script classes and is included here because the first two volumes in this series looked at other classes in that bundle. This is more complicated to use than letter but is more flexible. At the time of writing this, the current version of the scrlttr2 class is 3.14 (2014-10-28).

3.1 Writing a Letter Using the *letter* Class

- newlfm** This class is less flexible than `scrltr2` but can also be used for memos (see §6.1) and faxes. At the time of writing this, the current version of the `newlfm` class is dated 2009-04-11.
- isodoc** This class is more flexible than `newlfm` but less complicated than `scrltr2` and can also be used to write invoices (see §4.1). At the time of writing this, the current version of the `isodoc` class is 1.06 (2014-07-26).

⚠ Note that it's usually considered inappropriate to have floats within a letter, so letter-style classes, such as `letter`, `scrltr2` and `newlfm`, don't define the `figure` or `table` environments. You can, however, just use a `tabular` environment. In this case, you might want to place the tabular environment within the `center` environment to centre it and produce a small vertical gap above and below. The `isodoc` class inherits the `figure` and `table` environments from the underlying base `article` class, but that kind of floating material is liable to interfere with the fixed blocks such as the address information.

3.1 Writing a Letter Using the *letter* Class

This section describes how to use the basic `letter` class. As with all the base L^AT_EX classes, the paper size defaults to the US letter size, but this can be

3.1 Writing a Letter Using the *letter* Class

changed via the class options. For example, for A4 paper you can use the `a4paper` option:

```
\documentclass[a4paper]{letter}
```

Input

or you can use the `geometry` package [110]:

```
\documentclass{letter}  
\usepackage[a4paper]{geometry}
```

↑ Input

↓ Input

A document using the `letter` class may have one or more `letter` environments that contain the text of the letter:

```
\begin{letter}{<recipient's address>}  
<body>  
\end{letter}
```

Definition

Within `<body>`, you can use:

```
\opening{<salutation text>}
```

Definition

for the letter greeting, and

3.1 Writing a Letter Using the *letter* Class

\closing{<closing text>}

Definition

for the closing text (such as “Yours Sincerely”). After \closing you can also use:

\ps {<postscript text>}

Definition

for any postscripts (note that <postscript text> isn’t an argument and you need to supply your own “PS” tag at the start of it)

\cc{<text>}

Definition

to indicate a list of people to be cc’d, and

\encl{<text>}

Definition

to specify a list of any enclosures.

EXAMPLE 12. WRITING A SIMPLE LETTER (letter CLASS)

Here is a simple letter:

```
\documentclass[12pt]{letter}  
\usepackage[a4paper]{geometry}
```

↑ Input

3.1 Writing a Letter Using the *letter* Class

```
\usepackage[british]{babel}

\begin{document}

\begin{letter}{Mrs Mabel Canary\\24 The Street\\
Some Village\\Some Town\\Noshire\\AB1 2YZ}

\opening{Dear Mrs Canary}

This is an imaginary letter.

This is the second paragraph of the letter.

\closing{Yours sincerely}

\ps PS: this is a postscript.

\encl{Photocopy of something interesting\\
Photocopy of something rather dull}

\cc{Prof Important Person\\Dr Bor Ing}

\end{letter}
```

3.1 Writing a Letter Using the *letter* Class

```
\end{document}
```

↓ Input

(I've used the `babel` package [7] with the `british` option to ensure that the date is displayed using the British format rather than the default US format.)

The resulting document is shown in [Figure 3.1](#). You can [download](#) or [view](#) this example.

You can add information about the sender as well. This typically goes in the preamble. The sender's name is specified using:

```
\name{\langle text\rangle}
```

Definition

Additionally, you can specify the sender's name as it should appear after the closing text (supplied by `\closing`):

```
\signature{\langle text\rangle}
```

Definition

The sender's address is specified using:

3.1 Writing a Letter Using the letter Class



26th February 2014

Mrs Mabel Canary
24 The Street
Some Village
Some Town
Nowhere
AB1 2YZ

Dear Mrs Canary

This is an imaginary letter.

This is the second paragraph of the letter.

Yours sincerely

PS: this is a postscript.
enc: Photocopy of something interesting
Photocopy of something rather dull
cc: Prof Important Person
Dr Big Bag

Figure 3.1 A Simple Letter Using the letter Class

3.1 Writing a Letter Using the *letter* Class

`\address{<text>}`

Definition

You may use `\` within `<text>` to separate the lines of the address. The sender's telephone number is specified using:

`\telephone{<text>}`

Definition

Additional location information for the sender is specified using:

`\location{<text>}`

Definition

By default, the location and telephone number information is placed on the footer of the first page if no sender address has been specified. If you want to specify the sender address as well, you need to use

`\thispagestyle{firstpage}`

Input

after `\opening`.

The letter class is very old and doesn't provide a means for specifying modern communication methods such as email, mobile phone numbers or web addresses.

EXERCISE 7. WRITING A LETTER (letter CLASS)

Modify [Example 12](#) to include sender details.

3.2 Writing a Letter Using the `scrltr2` Class

FOR THE MORE ADVENTUROUS

Recall from §2.8 that you can fetch a single row of data from a database. Use one of the commands described in that section to fetch the recipient's details from the sample `people.csv` file or the `people` SQL table rather than explicitly typing them into the document. You can [download](#) or [view](#) a solution.

3.2 ■ Writing a Letter Using the `scrltr2` Class

This section describes how to use the `scrltr2` KOMA-Script class [47] to write letters. (Don't be confused if you see a class called `scrlettr`. That's an older deprecated class.)

The `scrltr2` class defines the `letter` environment:

```
\begin{letter}{<options>}{{<addressee>}}
```

Definition

where `<addressee>` is the recipient's postal details. Use `\\\` to separate the lines of the address. As with the `letter` class described [above](#), a document may have more than one `letter` environment, which will later come in useful when we look at mail merging in §3.5. Within the `letter` environment you must start the letter with

3.2 Writing a Letter Using the *scrltr2* Class

\opening{\langle salutation \rangle}

Definition

and close with

\closing{\langle sign-off text \rangle}

Definition

The *\langle salutation \rangle* is the greeting at the start of the letter, such as Dear Dr~Smith and the *\langle sign-off text \rangle* is the closing text, such as Yours sincerely.

After \closing{\langle sign-off text \rangle} you can optionally use:

\ps{\langle postscript text \rangle}

Definition

(As with the letter class, the *\langle postscript text \rangle* is not an argument of \ps, and you need to include “PS” in *\langle postscript text \rangle* if you want it as it’s not automatically generated.)

\encl{\langle enclosures info \rangle}

Definition

and

\cc{\langle cc list \rangle}

Definition

These three commands, if required, must go after \closing and before the end of the letter environment.

 The scrltr2 class is designed for use with the babel package [7], so remember to load it. (See Volume 1 [93, §5.8].)

3.2 Writing a Letter Using the *scrlttr2* Class

EXAMPLE 13. A SIMPLE LETTER (*scrlttr2* CLASS)

Here's a simple example letter:

```
\documentclass{scrlttr2}

\usepackage[british]{babel}

\begin{document}

\begin{letter}{Mrs Mabel Canary\\24 The Street\\Some Village\\
Some Town\\Noshire\\AB1 2YZ}
\opening{Dear Mrs~Canary}
```

↑ Input

This is an imaginary letter.

This is the second paragraph of the letter.

```
\closing{Yours sincerely}
```

```
\ps PS: this is a postscript.
```

3.2 Writing a Letter Using the *scrlttr2* Class

```
\encl{Photocopy of something interesting}\\
Photocopy of something rather dull}

\cc{Prof Important Person\\
Dr Bor Ing}
\end{letter}

\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.) The resulting document is shown in [Figure 3.2](#). Things to note:

- The first paragraph of the letter isn't indented, subsequent paragraphs are. If you prefer to have blank lines between paragraphs and no paragraph indentation you can use the `parskip=full` or `parskip=half` class options. For example:

```
\documentclass[parskip=full]{scrlttr2}
```

Input

- The date is automatically inserted. The date format is defined by the language dialect currently in use. Since I've loaded babel with the `british` option, the date is in the form `<day> <month> <year>`.

3.2 Writing a Letter Using the *scrlttr2* Class

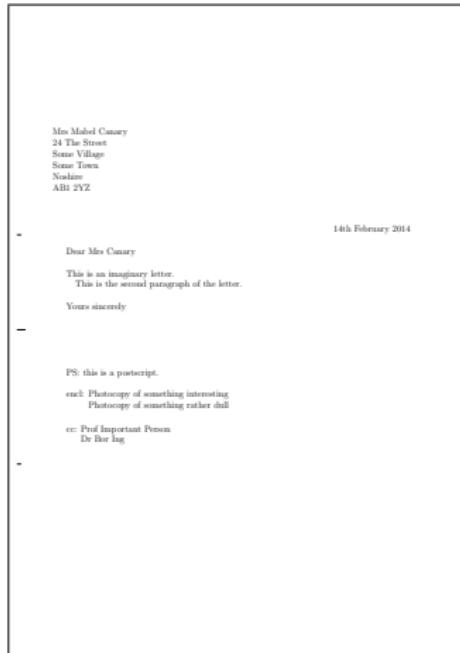


Figure 3.2 A Simple Letter Using the *scrlttr2* Class

3.2 Writing a Letter Using the *scrltr2* Class

- There is space between the closing text and the postscript for you to sign your name.
 - The page has horizontal marks on the left hand side. These are guides for folding or punching and are discussed below.
 - This example assumes that you are going to print the letter on headed paper.
-

KOMA-Script has “variables” that represents document elements. Variable names don’t start with a backslash. They are simple labels such as `yourref`. Variables have both content and a description. You can set the content, and optionally the description, using:

`\setkomavar{\langle name \rangle}{\langle description \rangle}{\langle content \rangle}`

Definition

where `\langle name \rangle` is the variable’s name, `\langle description \rangle` is the variable’s description and `\langle content \rangle` is the variable’s content. Alternatively, you can just set the description, without modifying the content, using:

`\setkomavar*{\langle name \rangle}{\langle description \rangle}`

Definition

3.2 Writing a Letter Using the *scrltr2* Class

EXAMPLE:

Suppose I want to add a “Your ref” line before the opening salutation. For example, supposing the recipient has requested the reference “ABC/123” in any correspondence, then I need to add

```
\setkomavar{yourref}{ABC/123}
```

Input

before the `\opening` command. This will now appear in the letter as



Your ref.
ABC/123

↑ Output

↓ Output

Here, the variable is `yourref` and the content is “ABC/123”. The default description (in English) for this variable is “Your ref.”. This can be changed with the optional argument:

```
\setkomavar{yourref}[Your Reference:]{ABC/123}
```

Input

KOMA-Script also has “options” that correspond to the variables. The option name is the same as the variable name and can be set via:

3.2 Writing a Letter Using the *scrltr2* Class

\KOMAoption{<option>}{<value list>}

Definition

or

\KOMAoptions{<option=value list>}

Definition

For example, the `subject` option indicates how the subject text should be displayed. Possible values for this option are: `afteropening`, `beforeopening`, `centered`, `left`, `right`, `titled`, `underlined` or `untitled`. For example, to make the subject appear after the opening text you can use either

\KOMAoptions{subject=afteropening}

Input

or

\KOMAoption{subject}{afteropening}

Input

Some of these settings can be combined. For example:

```
% set the subject options:  
\KOMAoption{subject}{afteropening,right,underlined,titled}  
% set the subject contents:  
\setkomavar{subject}{A sample letter}
```

↑ Input

↓ Input

3.2 Writing a Letter Using the *scrlttr2* Class

There are a lot of other variables, such as `invoice` for the invoice number and `customer` for the customer number. Common variables are listed in [Table 3.1](#). Common options with some of their possible values are listed in [Table 3.2](#). See the KOMA-Script documentation [47] for a complete list of all variables and options.

Options can also be set via the optional argument of the letter environment. For example:

```
\begin{letter}[subject=afteropening]{Mrs~Mabel Canary}
```

Input

or in the class options. For example:

```
\documentclass[subject=afteropening]{scrlttr2}
```

Input

The `foldmarks` option may have the single letter values combined. For example:

```
\KOMAoptions{foldmarks=vpmBT}
```

Input

or

```
\KOMAoption{foldmarks}{vpmBT}
```

Input

Table 3.1 Common *scr1tr2* Variables

Variable	Description
<code>customer</code>	Customer number.
<code>date</code>	Letter date.
<code>firstfoot</code>	Footer for the first page of the letter.
<code>nextfoot</code>	Footer for subsequent pages of the letter.
<code>firsthead</code>	Header for the first page of the letter.
<code>nexthead</code>	Header for subsequent pages of the letter.
<code>fromaddress</code>	Sender's address (without sender's name).
<code>fromemail</code>	Sender's email.
<code>fromfax</code>	Sender's fax number.
<code>fromlogo</code>	Commands for inserting sender's logo.
<code>frommobilephone</code>	Sender's mobile phone number.
<code>fromname</code>	Sender's full name.
<code>fromphone</code>	Sender's telephone number.
<code>fromurl</code>	Sender's URL (e.g. home page).
<code>invoice</code>	Invoice number.
<code>myref</code>	Sender's reference.
<code>location</code>	Additional sender details.
<code>signature</code>	Signature beneath letter ending.
<code>subject</code>	Letter's subject.

Common `scrLtr2` Variables (Continued)

Variable	Description
<code>title</code>	Letter's title.
<code>yourref</code>	Recipient's reference.

Table 3.2 Common `scrLtr2` Options with Some of their Values

Option	Value	Description
<code>firsthead</code>	<code>true</code>	Display letter head.
	<code>false</code>	Don't display letter head.
<code>fromalign</code>	<code>center</code>	Centre return address.
	<code>left</code>	Left-justify return address.
<code>fromphone</code>	<code>right</code>	Right-justify return address.
	<code>true</code>	Include sender's phone number.
<code>frommobilephone</code>	<code>false</code>	Don't include sender's phone number.
	<code>true</code>	Include sender's mobile number.
	<code>false</code>	Don't include sender's mobile number.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
<code>fromfax</code>	<code>true</code>	Include sender's fax number.
	<code>false</code>	Don't include sender's fax number.
<code>fromemail</code>	<code>true</code>	Include sender's email.
	<code>false</code>	Don't include sender's email.
<code>fromurl</code>	<code>true</code>	Include sender's web address.
	<code>false</code>	Don't include sender's web address.
<code>fromlogo</code>	<code>true</code>	Include sender's logo.
	<code>false</code>	Don't include sender's logo.
<code>addrfield</code>	<code>true</code>	Print an address field including a return address, mode of dispatch and priority.
<code>backaddress</code>	<code>false</code>	Don't print address field.
	<code>true</code>	Print return address for window envelope.
	<code>false</code>	Don't print return address for window envelope.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
priority	false	Don't print priority field.
	economy	Use international priority B-Economy.
	priority	Use international priority A-Priority.
locfield	narrow	Narrow location field.
	wide	Wide location field.
numericaldate	true	Use numerical date.
	false	Use language-dependent date format.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
<code>refline</code>	<code>dateleft</code>	Place date left-most on the reference line.
	<code>dateright</code>	Place date right-most on the reference line.
	<code>narrow</code>	Restrict reference line to <code>typearea</code> .
	<code>nodate</code>	Don't place date on reference line.
	<code>wide</code>	The width of the reference line corresponds to the address and sender's additional details.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
subject	true	Print subject field.
	afteropening	Place subject field below letter opening.
	beforeopening	Place subject field above letter opening.
	centered	Centre subject field.
	left	Left-justify subject field.
	right	Right-justify subject field.
	title	Add title/description to subject field.
	untitled	Don't add title/description to subject field.
	underlined	Underline subject field (must be single-lined).
headsepline	true	Insert a separator line below the header.
	false	Don't insert a separator line below the header.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
footsepline	true	Insert a separator line above the footer.
	false	Don't insert a separator line above the footer.
pagenumber	foot	Page number in footer.
	head	Page number in header.
	false	No page number.
	center	Page number centred.
	left	Page number left-aligned.
	right	Page number right-aligned.

Common `scrltr2` Options with Some of their Values (Continued)

Option	Value	Description
<code>foldmarks</code>	<code>false</code>	Don't display fold marks.
	<code>B</code>	Activate upper horizontal foldmark on left paper edge.
	<code>b</code>	Deactivate upper horizontal foldmark on left paper edge.
	<code>H</code>	Activate all horizontal folding marks on left paper edge.
	<code>h</code>	Deactivate all horizontal folding marks on left paper edge.
	<code>L</code>	Activate left vertical foldmark on upper paper edge.
	<code>l</code>	Deactivate left vertical foldmark on upper paper edge.
	<code>M</code>	Activate middle horizontal foldmark on left paper edge.
	<code>m</code>	Deactivate middle horizontal foldmark on left paper edge.
	<code>P</code>	Activate punch or centre mark on left paper edge.

3.2 Writing a Letter Using the *scrltr2* Class

Common *scrltr2* Options with Some of their Values (Continued)

Option	Value	Description
	p	Deactivate punch or centre mark on left paper edge.
	T	Activate lower horizontal foldmark on left paper edge.
	t	Deactivate lower horizontal foldmark on left paper edge.
	V	Activate all vertical folding marks on upper paper edge.
	v	Deactivate all vertical folding marks on upper paper edge.

Recall from Volume 1 [93, §6] that the *graphicx* package [14] provides the command:

`\includegraphics[<options>]{<file>}`

Definition

to include the image file called *<file>*. (The extension may be omitted.)

By default, the logo isn't displayed, even if you set the *fromlogo* variable. If you want it displayed you need to activate it via:

3.2 Writing a Letter Using the *scrlttr2* Class

```
\KOMAoption{fromlogo}{true}
```

Input

as well as set the `fromlogo` variable:

```
\setkomavar{fromlogo}{\includegraphics{mylogo}}
```

Input

EXAMPLE 14. WRITING A LETTER: KOMA SETTINGS

Remember that you can have multiple letters in the same document. If you want the content of a variable to be the same for all the letters in the document, put the `\setkomavar` or `\setkomavar*` command in the preamble. If the content of a variable depends on the recipient, then set it inside the `letter` environment (before `\opening`), as illustrated below:

```
\documentclass[12pt,parskip=full]{scrlttr2}

\usepackage[british]{babel}

\KOMAoption{subject}{afteropening,right,underlined,titled}

\KOMAoptions{foldmarks=vpmBT}
```

↑ Input

3.2 Writing a Letter Using the *scrlttr2* Class

```
\setkomavar{signature}{Mr Big Head, Managing Director}
\setkomavar{subject}{A sample letter}

\begin{document}

\begin{letter}{Mrs Mabel Canary\\24 The Street\\Some Village\\
Some Town\\Noshire\\AB1 2YZ}
\setkomavar{myref}{ABC/123}
\setkomavar{invoice}{123456}
\setkomavar{customer}{2}

\opening{Dear Mrs~Canary}

This is an imaginary letter.

This is the second paragraph of the letter.

\closing{Yours sincerely}

\ps PS: this is a postscript.

\encl{Photocopy of something interesting\\}
```

3.2 Writing a Letter Using the *scrltr2* Class

Photocopy of something rather dull}

```
\cc{Prof Important Person}\\Dr Bor Ing}  
\end{letter}  
  
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 3.3](#).

EXERCISE 8. WRITING A LETTER (*scrltr2* CLASS)

For this exercise, modify the sample letter from [Example 14](#) so that it includes a return address and logo. You can make up the sender's address or use your own. If you don't have an image file, you can download the sample logo [dummy-logo.png](#). Alternatively you can use the `example-image.pdf` file included with the `mwe` package [83]. (You don't need to load `mwe` in order to use this image.) Remember you can scale the image using the optional argument of `\includegraphics`. Also, try changing some of the options, such as switching off all the fold and punch marks. You can [download](#) or [view](#) a solution.

3.2 Writing a Letter Using the *scrlltr2* Class

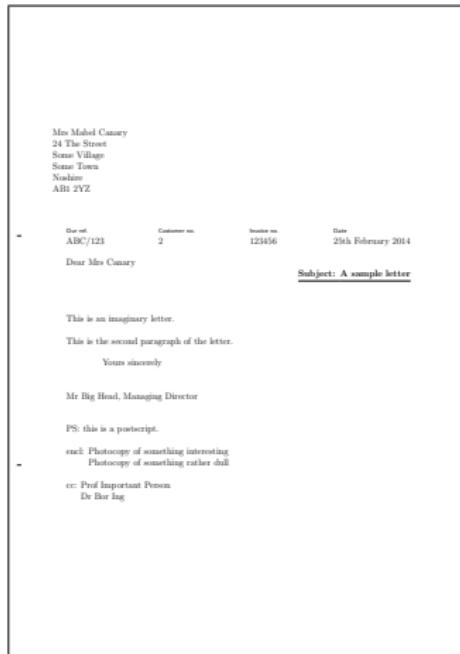


Figure 3.3 A Simple Letter Using KOMA-Script Variables

3.3 Writing a Letter Using the `newlfm` Class

FOR THE MORE ADVENTUROUS

Recall from §2.8 that you can fetch a single row of data from a database. Use one of the commands described in that section to fetch the recipient's details from the sample `people.csv` file or the `people` SQL table rather than explicitly typing them into the document. You can [download](#) or [view](#) a solution.

3.3 ■ Writing a Letter Using the `newlfm` Class

The `newlfm` class [107] provides the `newlfm` environment

```
\begin{newlfm}  
(text)  
\end{newlfm}
```

Definition

where `(text)` is the body of the letter. The salutation text and recipient's name and address all need to be specified before this environment.

The date of the letter defaults to the current date (via `\today`) but can be set using:

3.3 Writing a Letter Using the *newlfm* Class

`\dateset{(date)}`

Definition

The salutation text is specified via:

`\greetto{(text)}`

Definition

For example

`\greetto{Dear Mrs Canary}`

Input

The closing text is specified via:

`\closeline{(text)}`

Definition

For example

`\closeline{Yours sincerely}`

Input

The recipient's name is specified via:

`\nameto{(name)}`

Definition

For example:

`\nameto{Mrs Mabel Canary}`

Input

The recipient's address is specified via:

3.3 Writing a Letter Using the *newlfm* Class

\addrto{<address>}

Definition

For example:

```
\addrto{24 The Street\\
Some Village\\
Some Town\\
Noshire\\
AB1 2YZ
}
```

↑ Input

↓ Input

The sender's name is specified via:

\namefrom{<name>}

Definition

For example:

```
\namefrom{Mr Big Head}
```

Input

The sender's address is specified via:

3.3 Writing a Letter Using the *newlfm* Class

\addrfrom{<address>}

Definition

For example

```
\addrfrom{University of Somewhere\\
Some City\\
AB3 4YZ}
```

↑ Input

↓ Input

The sender's phone number is specified via:

\phonefrom{<number>}

Definition

For example

```
\phonefrom{@123456789}
```

Input

The sender's email address is specified via:

\emailfrom{<address>}

Definition

For example

3.3 Writing a Letter Using the *newlfm* Class

\emailfrom{big.head@somewhere.ac.uk}

Input

The subject text is specified via:

\regarding{\langle text\rangle}

Definition

For example:

\regarding{sample letter}

Input

The postscript is specified via:

\psitem{\langle text\rangle}

Definition

For example:

\psitem{Don't forget to bring some cake!}

Input

The post-postscript is specified via:

\ppsitem{\langle text\rangle}

Definition

For example:

\ppsitem{And the ice cream!}

Input

The “CC” list is specified via:

3.3 Writing a Letter Using the `newlfm` Class

`\cclist{<text>}`

Definition

For example:

`\cclist{Prof Important Person}`

Input

The list of enclosures is specified via:

`\enclist{<text>}`

Definition

For example:

`\enclist{Photograph of a hat}`

Input

There are other commands as well. See the `newlfm` documentation [107] for further details.

The `newlfm` class comes with a number of predefined letter styles, which can be set via the class options: `busletter`, `busletternofrom`, `stdletter`, `stdletterfrom`. The “`nofrom`” styles don’t display the sender’s address. The business letter “`busletter`” styles use a different alignment to the standard “`stdletter`” styles.

Alternatively, the letter style can be set via:

`\newlfmP{<option list>}`

Definition

Other letter-related options that can be set using `\newlfmP` (or in the class option list) are listed in [Table 3.3](#).

Table 3.3 Letter Options for the `newlfm` Class

Option	Description
<code>noaddrfrom</code>	Omit sender's address.
<code>addrfromphone</code>	Include sender's phone.
<code>addrfromemail</code>	Include sender's email.
<code>addrfromfax</code>	Include sender's fax.
<code>printallfrom</code>	Print all of the sender's details.
<code>addrfromright</code>	Right-align sender's block.
<code>addrfromleft</code>	Left-align sender's block.
<code>printallto</code>	Print all of recipient's details.
<code>addrtright</code>	Right-align recipient's block.
<code>addrtoleft</code>	Left-align recipient's block.
<code>addrtophone</code>	Include recipient's phone.
<code>addrtoemail</code>	Include recipient's email.
<code>addrtofax</code>	Include recipient's fax.
<code>dateright</code>	Right-align date.
<code>dateleft</code>	Left-align date.

3.3 Writing a Letter Using the `newlfm` Class

Table 3.3 Letter Options for the `newlfm` Class (Continued)

Option	Description
<code>datecenter</code>	Centre date.
<code>dateyes</code>	Include date.
<code>dateno</code>	Don't include date.
<code>orderdatefromto</code>	Display order: date, sender, recipient.
<code>orderfromtodate</code>	Display order: sender, recipient, date.
<code>orderfromdateto</code>	Display order: sender, date, recipient.
<code>sigright</code>	Right-align signature.
<code>sigleft</code>	Left-align signature.
<code>sigcenter</code>	Centre signature.

EXAMPLE 15. A SIMPLE LETTER (`newlfm` CLASS)

Here's a simple letter using the `newlfm` class:

```
\documentclass[stdletter]{newlfm}
```

↑ Input

3.3 Writing a Letter Using the *newlfm* Class

```
\usepackage[british]{babel}

\newlfmP{orderfromtodate,sigcenter,addrfromphone,addrfromemail}

\nameto{Mrs Mabel Canary}
\addrto{24 The Street\\
Some Village\\Some Town\\
Noshire\\AB1 2YZ}

\namefrom{Mr Big Head}
\addrfrom{University of Somewhere\\Some City\\AB3 4YZ}
\emailfrom{big.head@somewhere.ac.uk}
\phonefrom{0123456789}

\regarding{A sample letter}

\greetto{Dear Mrs Canary}
\closeline{Yours sincerely}
\cclist{Prof Important Person\\Dr Bor Ing}
\encllist{Photocopy of something interesting\\
Photocopy of something rather dull}
\psitem{this is a postscript}
```

3.3 Writing a Letter Using the `newlfm` Class

```
\begin{document}
```

```
\begin{newlfm}
```

This is an imaginary letter.

This is the second paragraph of the letter.

```
\end{newlfm}
```

```
\end{document}
```

Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 3.4](#).

EXERCISE 9. WRITING A LETTER (`newlfm` CLASS)

Create the document from [Example 15](#) and try adjusting some of the options given in [Table 3.3](#).

FOR THE MORE ADVENTUROUS

As with [Exercise 8](#), use one of the commands described in [§2.8](#) to fetch the recipient's details from the sample `people.csv` file or the `people` SQL table

3.3 Writing a Letter Using the *newlfm* Class



Figure 3.4 A Simple Letter Using the *newlfm* Class

3.4 Writing a Letter Using the *isodoc* Class

rather than explicitly typing them into the document. You can [download](#) or [view](#) a solution.

3.4 □ Writing a Letter Using the *isodoc* Class

The *isodoc* class [21] differs from the classes described in the previous sections in that instead of encasing the letter within the body of an environment (such as `letter` or `newlfm`) the letter is created using a command:

`\letter[<recipient options>]{<contents>}`

Definition

As with the other classes, a document may contain multiple letters. The optional argument `<recipient options>` is a [key=value list](#) of options to apply to this letter. The other argument `<contents>` contains the contents of the letter. General options can be set using:

`\setupdocument{<options>}`

Definition

Again, `<options>` is a [key=value list](#). There are a lot of options available, so this section will only cover common settings. For full details, see the *isodoc* user guide [21]

3.4 Writing a Letter Using the *isodoc* Class

You don't need to use babel with isodoc. Instead you can set the language using the language key. As of the time of writing, available language options are: en-GB (default), en-US, fr-FR, de-DE, nl-NL, nl-BE, it-IT, es-ES, ca-ES, nb-NO and sr-RS. The hyphen is optional so, for example, enGB is the same as en-GB. For example, to switch to US English:

```
\setupdocument{language=en-US}
```

Input

Options that set information about the sender include:

- | | |
|-------------|---|
| company | Company name (or sender's name if a private document). |
| logoaddress | Sender's address. If omitted it will be constructed from the following: |
| who | Contact person's name. |
| street | Sender's street. |
| city | Sender's city. |
| zip | Sender's zip or postcode. |
| countrycode | Sender's country code. |

*3.4 Writing a Letter Using the *isodoc* Class*

country	Sender's country name.
areacode	Sender's area code.
cityzip	Place the zip code after the city.
foreign	If this key is used, the country name will be added to the address, the zip code will be prefixed with the country code and the telephone numbers will be prefixed with the area code. (This isn't a boolean key , but if you want to generate multiple letters with a mixture of national and international addresses, you can switch this setting on and off using \foreigntrue and \foreignfalse.)

The above information is placed in the logo area. The options that govern the address window include:

to	The recipient's address. You can use \\\ to break the lines.
return	Include the return address in the address window.
returnaddress	If the return address is too long for the address window, this key can be used to set a shorter version for the window.

3.4 Writing a Letter Using the *isodoc* Class

Options that set the header information include:

yourletter If this letter is a reply to a letter from the recipient, this value is the date of the recipient's letter.

yourref The recipient's reference if this is a reply.

ourref The sender's reference.

date The date of this letter, which must be in the form `<yyyy>-<mm>-<dd>` or `<yyyy><mm><dd>`. This will be converted into the format governed by the language setting (as specified by the language key described above). For example:

`\setupdocument{date={2014-03-01}}`

Input

indicates the first day of March, 2014.

forcedate May be used instead of the previous key to force the data to be in a specific format. For example:

3.4 Writing a Letter Using the *isodoc* Class

```
\setupdocument{forcedate={Sat 1st March, 2014}}
```

Input

subject The subject of this letter.

Options that set the opening and closing information include:

opening The opening salutation.

closing The closing salutation.

signature The name of the sender as it should appear below the closing salutation.

enclosures Lists any enclosures accompanying the letter (may include \\ to start a newline).

copyto A "CC" list. Again this may include \\.

autograph This option governs the area between the closing text and the signature text. The value may be one of:

- 0 No space between the closing and signature text (default);
- 1 Leaves a space between the closing and signature text for a handwritten signature;

3.4 Writing a Letter Using the *isodoc* Class

- 2–9** Inserts one of eight autograph images (see the *isodoc* manual [21] for further details).

Options that set the footer information include:

`footer` Enables the footer information.

`phoneprefix` Sets the phone prefix (defaults to 0).

`phone` The sender's phone number (omit the phone prefix).

`cellphone` The sender's mobile phone (omit the phone prefix).

`fax` The sender's fax number (omit the phone prefix).

`email` The sender's email address.

`website` The sender's web address.

There are other options that govern the layout. See the *isodoc* documentation [21] for further details. There are also other options that are concerned with invoices. These are described in §4.1.

3.4 Writing a Letter Using the *isodoc* Class

EXAMPLE 16. A SIMPLE LETTER (isodoc CLASS)

The letter from Example 14 can be rewritten using the *isodoc* class as follows:

```
\documentclass[12pt]{isodoc}

\setupdocument
{%
language={en-GB},%
company={University of Somewhere},%
who={Mr Big Head},%
street={Academic Lane},%
city={Some City},%
zip={AB3 4YZ},%
country={United Kingdom},%
countrycode={GB},%
areacode={44},%
cityzip,%
subject={A sample letter},%
closing={Yours sincerely},%
enclosures={Photocopy of something interesting}\\}
```

↑ Input

3.4 Writing a Letter Using the *isodoc* Class

```
Photocopy of something rather dull},%
signature={Big Head},%
copyto={Prof Important Person\\Dr Bor Ing},%
footer,%
phone={123456789},%
cellphone={712345678},%
email={big.head@somewhere.ac.uk},%
website={somewhere.ac.uk},%
date={2014-03-01}%
}

\begin{document}

\letter
[% 
opening={Dear Mrs Canary},%
to={Mrs Mabel Canary\\%
24 The Street\\%
Some Village\\Some Town\\%
Noshire\\AB1 2YZ},%
ourref={ABC/123}%
]%
```

3.5 Mail Merging

```
{%
```

```
  This is an imaginary letter.
```

```
  This is the second paragraph of the letter.
```

```
}
```

```
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 3.5](#). There's no space between the closing text and signature as I didn't use the autograph option.

3.5 Mail Merging

The `scrletter2` and `newlfm` classes both provide ways of creating a template letter for mail merging. However, in case you don't want to be fixed to a specific class, or you want to use the `letter` or `isodoc` class (or some other class not described in this book), this section looks at a more generic method of mail-merging using the `datatool` package.

3.5 Mail Merging

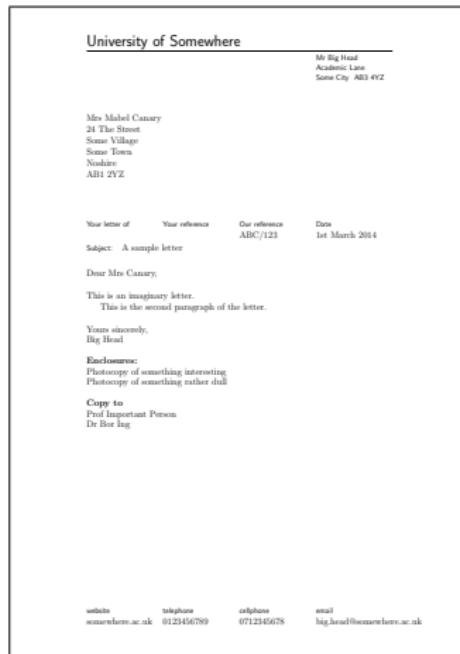


Figure 3.5 A Simple Letter Using the `isodoc` Class

3.5 Mail Merging

Recall from §2.7.1 that you can iterate through a datatool database using `\DTLforeach*`. This technique can be used to create a letter for each person in the database, or you can apply filtering to only send to a subset of the database.

In fact, this is just a small modification of the “For the More Adventurous” sections of [Exercise 7](#), [Exercise 8](#) and [Exercise 9](#). Any settings, such as the sender’s details, that stay constant for all the letters can be set before `\DTLforeach*`. The `letter` or `newlfm` environment can go inside the body of `\DTLforeach*`.

EXAMPLE 17. MAIL MERGING (letter CLASS)

To send a letter to everyone listed in sample `people.csv` file:

```
\documentclass[letter]

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[a4paper]{geometry}
\usepackage[british]{babel}
\usepackage{datatool}
```

↑ Input

3.5 Mail Merging

```
\DTLloaddb{people}{people.csv}
\DTLloaddb{countries}{country-codes.csv}

\name{Mr Big Head}
\signature{Big Head}
\location{Secret Lab of Experimental Stuff}
\address{University of Somewhere\\Some City\\AB3 4YZ}
\telephone{0123456789}

\begin{document}

\DTLforeach*{people}{% data
{%
assignments
\Id=id,%
\Surname=surname,%
\Forenames=forenames,%
>Title=title,%
\AddressI=address1,%
\AddressII=address2,%
\Town=town,%
\County=county,%
}}
```

3.5 Mail Merging

```
\Postcode=postcode,%
\CountryCode=country%
}
{%
% fetch country name
\xDTLassignfirstmatch{countries}{code}{\CountryCode}{\CountryName=name}

\begin{letter}{% recipient's address
\DTLifnullorempty{\Title}{}{\Title \l}\Forenames \l \Surname \\%
\AddressI \\
\DTLifnullorempty{\AddressII}{}{\AddressII \\}% optional line
\Town \\
\DTLifnullorempty{\County}{}{\County \\}% optional line
\Postcode \\ \CountryName}

\opening{Dear \DTLifnullorempty{\Title}{\Forenames}{\Title} \Surname}

\thispagestyle{firstpage}
This is an imaginary letter.

This is the second paragraph of the letter.
```

3.5 Mail Merging

```
\closing{Yours sincerely}

\ps PS: this is a postscript.

\encl{Photocopy of something interesting\\
Photocopy of something rather dull}

\cc{Prof Important Person\\Dr Bor Ing}
\end{letter}

}

\end{document}
```

↓ Input

This produces a six page document, where each page contains a letter to one of the six people in the database. You can [download](#) or [view](#) this document.

EXAMPLE 18. MAIL MERGING (newlfm CLASS)

To send a letter to everyone listed in sample [people.csv](#) file:

3.5 Mail Merging

↑ Input

```
\documentclass[stdletter]{newlfm}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[british]{babel}
\usepackage{datatool}

\DTLloaddb{people}{people.csv}
\DTLloaddb{countries}{country-codes.csv}

\newlfmP{orderfromtodate,sigcenter,addrfromphone,addrfromemail}

\namefrom{Mr Big Head}
\addrfrom{University of Somewhere\\Some City\\AB3 4YZ}
\emailfrom{big.head@somewhere.ac.uk}
\phonefrom{0123456789}

\regarding{A sample letter}
```

3.5 Mail Merging

```
\begin{document}

\closeline{Yours sincerely}

\cclist{Prof Important Person\\Dr Bor Ing}

\encllist{Photocopy of something interesting\\
Photocopy of something rather dull}

\psitem{this is a postscript}

\DTLforeach*{people}{%
  % data
  % assignments
  \Id=id,
  \Surname=surname,
  \Forenames=forenames,
  \Title=title,
  \AddressI=address1,
  \AddressII=address2,
  \Town=town,
  \County=county,
  \Postcode=postcode,
}
```

3.5 Mail Merging

```
\CountryCode=country%
}
{%
\xDTLassignfirstmatch{countries}{code}{\CountryCode}{\CountryName=name}

\nameto{\DTLifnullorempty{\Title}{}{\Title \_}\Forenames \_ \Surname}
\addrto{%
  \AddressI \\
  \DTLifnullorempty{\AddressII}{}{\AddressII \\}% optional line
  \Town \\
  \DTLifnullorempty{\County}{}{\County \\}% optional line
  \Postcode \\ \CountryName
}

\greetto{Dear \DTLifnullorempty{\Title}{\Forenames}{\Title} \Surname}

\begin{newlfm}
  This is an imaginary letter.

  This is the second paragraph of the letter.
\end{newlfm}
}
```

3.5 Mail Merging

```
\end{document}
```

↓ Input

As with the previous example, this produces a six page document, where each page contains a letter to one of the six people in the database. You can [download](#) or [view](#) this document.

NOTE

The `newlfm` environment tries to determine the total number of pages per letter. This is done using

```
\label{totpage}
```

Input

at the end of the letter. Since there are six letters, this causes five instances of

LaTeX Warning: Label `totpage' multiply defined.

However, since each letter is of the same length, these warnings can be ignored.



EXERCISE 10. MAIL MERGING

Adapt the more adventurous section of [Exercise 8](#) so that a letter is generated for each person in the sample `people.csv` file who has the subscribed field set. If you prefer, you can use the `people` SQL table. (Recall the `\ifcsbool` command defined on page [212](#). Alternatively you can use the optional argument of `\DTLforeach*` with ifthen's `\equal` command.)

The recipient's country only needs to be included if it's different from the sender's country. For example, suppose the sender's address is in the United Kingdom, then the country name isn't required for the entries where the recipient's country code is "gb". Modify your code so that it doesn't include the country name in the recipient's address if it's the same as the sender's country. You can [download](#) or [view](#) a solution.

If you prefer, you can use one of the other letter classes, such as `isodoc`, described in [§3.4](#). You can [download](#) or [view](#) a solution using the `isodoc` class.

3.6 Envelopes

The letter class defines a preamble-only command:

3.6 Envelopes

`\makelabels`

Definition

which gathers the addresses of all the recipients and generates address labels at the end of the document. These labels can then be stuck onto the envelopes.

For example, the document from [Example 17](#) had six pages, one page for each letter. If you add `\makelabels` to the preamble of that document, a seventh sheet will be generated with the address labels, shown in [Figure 3.6](#).

This is fine if this matches the size of your label sheets, but it can't be easily adapted for other label sizes. There are some envelope-related packages listed on the \TeX Catalogue Topic Index [25] or at the [letter topic](#) but take care as some of them, such as `envelope`, were written for $\text{\LaTeX}2.09$ and may not work as well with $\text{\LaTeX}2\epsilon$ or licensing issues may prevent them from being included in $\text{\TeX} \text{ Live}$.

Some of the other letter-like classes also provide envelope labelling facilities, but there are two packages that are both in the $\text{\TeX} \text{ Live}$ and MiK \TeX distributions: `envelab` and `envbig`. Another possibility is to use the `ticket` package, which is discussed in [§10.2](#).

The user guide for `envelab` [111] can be obtained via:

```
texdoc elguide
```

Shell

3.6 Envelopes

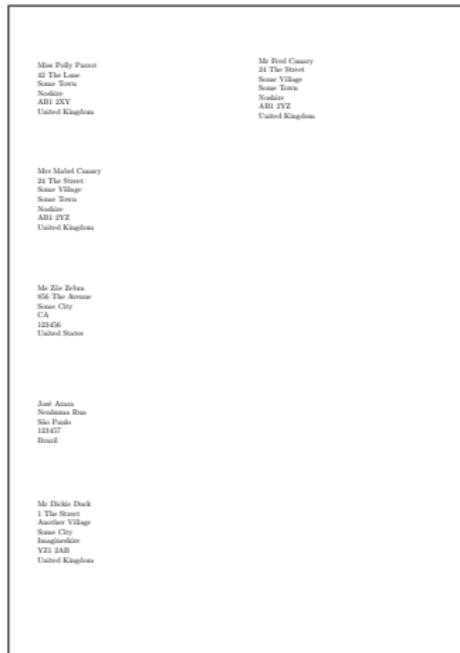


Figure 3.6 Address Labels

3.6 Envelopes

Note that `texdoc envlab` produces the documented source code, which you may find more complicated than the user guide. There's no proper manual for `envbig`; `texdoc envbig` just opens the README file. You need to open `envbig.sty` in your text editor and read the comments for an example of how to use the package. Therefore this book will just look at `envlab`. At the time of writing, the current version of `envlab` is 1.2 (1997-07-16).

The `envlab` package is designed for US postal layouts, but it's possible to define custom label sizes. This package redefines `\makelabels` but it's used in the same way as for the `letter` class.

The `envlab` package is configured for three different types of media: envelopes (one per page, optionally rotated), labels (without a return address) and big labels (with a return address). Envelopes are usually printed in landscape format. This rotation can be switched on or off using the `rotateenvelopes` or `norotateenvelopes` options.

The envelope layout can be set by the package options listed in [Table 3.4](#) or a custom size can be set via:

`\SetEnvelope[<top margin>]{<width>}{<height>}`

[Definition](#)

where `<top margin>` is the height of the top margin, `<width>` is the envelope width and `<height>` is the envelope height.

The standard label layout can be set by the package options listed in [Table 3.5](#) or a custom size can be set via:

3.6 Envelopes

Table 3.4 Envelope Options for the `envelab` Package

Option	Width	Height
<code>businessenvelope</code>	9.5 in	4.125 in
<code>executiveenvelope</code>	7.5 in	3.875 in
<code>bookletenvelope</code>	10.5 in	7.5 in
<code>personalenvelope</code>	6.5 in	3.625 in
<code>c6envelope</code>	162 mm	114 mm
<code>c65envelope</code>	224 mm	114 mm
<code>c5envelope</code>	229 mm	162 mm
<code>dlenvelope</code>	220 mm	110 mm

```
\SetLabel{<width>}{<height>}{<top>}{<left>}{<sep>}{<columns>}{<rows>}
```

Definition

where $\langle width \rangle$ is the total width from the left border of one label and the left border of the label in the next column, $\langle height \rangle$ is the total height from the top border of one label and the top border of the label in the row below, $\langle top \rangle$ and $\langle left \rangle$ are the distances between the edge of the paper and the label, $\langle sep \rangle$ is the horizontal distance between labels and $\langle columns \rangle$ and $\langle rows \rangle$ are the number of columns and rows of labels per page.

The big label layout can be set by the package options listed in Table 3.6

3.6 Envelopes

Table 3.5 Standard Label Options for the `envelab` Package

Option	Width	Height	Top	Left	Sep	Cols	Rows
<code>avery5160label</code>	2.75 in	1 in	0.5 in	0.19 in	0.12 in	3	10
<code>avery5161label</code>	4.19 in	1 in	0.5 in	0.16 in	0.19 in	2	10
<code>avery5162label</code>	4.19 in	1.33 in	0.83 in	0.16 in	0.19 in	2	7
<code>avery5163label</code>	4.19 in	2 in	0.5 in	0.16 in	0.19 in	2	5
<code>avery5164label</code>	4.19 in	3.33 in	0.5 in	0.16 in	0.19 in	2	3
<code>avery5262label</code>	110 mm	34 mm	21 mm	4 mm	5 mm	2	7
<code>herma4625label</code>	105 mm	42.3 mm	0 mm	5 mm	5 mm	2	7

or a custom size can be set via:

`\SetBigLabel{<width>}{<height>}{<top>}{<left>}{<sep>}{<columns>}
{<rows>}`

Definition

The arguments are the same as for `\SetLabel`.

If you have a partially used sheet, you can specify the starting label via:

`\FirstLabel{<row>}{<column>}`

Definition

where $\langle \text{row} \rangle$ is the row index (starting from 1) and $\langle \text{column} \rangle$ is the column index (starting from 1). The labels are printed row by row.

3.6 Envelopes

Table 3.6 Big Label Options for the `envelab` Package

Option	Width	Height	Top	Left	Sep	Cols	Rows
<code>avery5163biglabel</code>	4.19 in	2 in	0.5 in	0.16 in	0.19 in	2	5
<code>avery5164biglabel</code>	4.19 in	3.33 in	0.5 in	0.16 in	0.19 in	2	3

The return address for the big envelopes is taken from the argument of `\address` but this can be changed by redefining:

`\returnaddress`

Definition

This can be changed to the textual address or it can use `\includegraphics` to use a company logo.

EXAMPLE 19. ENVELOPE LABELS

This example creates a letter using the `letter` class but uses the `envelab` package to create a custom sized big label.

```
\documentclass{letter}  
  
\usepackage[a4paper]{geometry}
```

↑ Input

3.6 Envelopes

```
\usepackage[british]{babel}

\usepackage{envlab}

\SetBigLabel{101mm}{139mm}{9mm}{3mm}{2mm}{2}{2}

\makelabels

\name{Mr Big Head}
\signature{Big Head}
\location{Secret Lab of Experimental Stuff}
\address{University of Somewhere\\Some City\\AB3 4YZ}
\telephone{0123456789}

\begin{document}

\begin{letter}{Miss Polly Parrot\\42 The Lane\\Some Town\\AB1 2XY}

\opening{Dear Miss Parrot}

\thispagestyle{firstpage}
This is an imaginary letter.
```

3.6 Envelopes

This is the second paragraph of the letter.

```
\closing{Yours sincerely}
```

```
\ps PS: this is a postscript.
```

```
\encl{Photocopy of something interesting\\  
Photocopy of something rather dull}
```

```
\cc{Prof Important Person\\Dr Bor Ing}  
\end{letter}
```

```
\end{document}
```

↓ Input

This produces a document with two pages. The first contains the letter and the second contains the label. This second page is shown in [Figure 3.7](#).

This produces a portrait label, but if you have a wide address it may look better rotated to make a landscape label. Unfortunately, the `envelope` options that govern rotation (`rotateenvelope` and `norotateenvelope`) don't apply to labels.

These big labels are implicitly typeset using

3.6 Envelopes



Figure 3.7 Custom Big Label

3.6 Envelopes

\PrintBigLabel{<from-address>}{<to-address>}

Definition

where <from-address> is the sender's address and <to-address> is the recipient's address. This command is defined as:

```
\newcommand{\PrintBigLabel}[2]{%
\begin{minipage}[t][\LabelHeight]{\LabelWidth}%
\baselineskip=0pt%
\lineskip=0pt%
\parindent=0pt%
\begin{center}%
\PrintReturnAddress{#1}%
\rule{\ToAddressWidth}{0.1pt}%
\PrintAddress{#2}%
\end{center}%
\end{minipage}}
```

↑ Input

↓ Input

This displays the label in a `minipage` with the dimensions given by `\LabelWidth` and `\LabelHeight` which have been set to the width and height of the printable label area. This can be redefined to provide your own custom label

3.6 Envelopes

format. For example (recall `\rotatebox` provided by the `graphicx` package [14] described in Volume 1 [93, §6.1]):

```
\renewcommand{\PrintBigLabel}[2]{%
  \begin{minipage}[t]{\LabelWidth}{\LabelHeight}
    \vfill
    \hspace*{1em}%
    \rotatebox[origin=l]{90}{\PrintReturnAddress{\#1}}\hfill
    \rule{0.1pt}{\ToAddressWidth}\space
    \rotatebox[origin=l]{90}{\PrintAddress{\#2}}%
    \hspace*{1em}%
    \vfill
  \end{minipage}
}
```

↑ Input

↓ Input

This now produces the label shown in Figure 3.8. You can further customize this redefinition if you like. For example, you may want the recipient's address above the return address. You can [download](#) or [view](#) this example document.

3.6 Envelopes



Figure 3.8 Custom Formatted Big Label

3.6 Envelopes

⚠ Unfortunately the default `capaddress` package option doesn't work with **extended characters** unless they have been placed inside a group, which will cause a problem with some of the entries in the sample `people.csv` file or the `people` SQL table. This means that if we want to adapt [Example 17](#) to use `envelab`, we have to switch off the capitalisation feature using the `nocapaddress` package option.

EXAMPLE:

```
\documentclass{letter}
```

↑ Input

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

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

```
\usepackage[nocapaddress]{envelab}
```

```
\makelabels
```

```
\begin{document}
```

```
\begin{letter}{Ms Zöe Zebra\\856 The Avenue}
```

3.6 Envelopes

```
\opening{Dear Miss Parrot}
```

A sample letter.

```
\closing{Yours sincerely}
```

```
\end{letter}
```

```
\end{document}
```

↓ Input

With the `nocapaddress` option, this code compiles without error. With the default `capaddress` option, the ö must be placed inside a group:

```
\begin{letter}{Ms Z{ö}e Zebra\\856 The Avenue}
```

Input

(For those of you who have used glossaries or `mfirstuc`, it stems from a similar issue. See the section “UTF-8” in the `mfirstuc` documentation [99].)

Alternatively use `XeLaTeX` instead of `PDFLaTeX` (the `inputenc` and `fontenc` packages need to be replaced by the `fontspec` [76] package):

```
% arara: xelatex
```

↑ Input

3.6 Envelopes

```
\documentclass{letter}

\usepackage{fontspec}
\usepackage{environ}

\makelabels

\begin{document}

\begin{letter}{Ms Zöe Zebra \\\ 856 The Avenue}

\opening{Dear Miss Parrot}

A sample letter.

\closing{Yours sincerely}

\end{letter}

\end{document}
```

↓ Input

3.6 Envelopes

EXAMPLE 20. MAIL MERGING WITH letter AND envlab

The above can be put together to form a complete document that contains the correspondence and the large mailing labels:

↑ Input

```
\documentclass{letter}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[a4paper]{geometry}
\usepackage[british]{babel}
\usepackage{datatool}
\usepackage{graphicx}

\usepackage[nocapaddress]{envlab}

\DTLloaddb{people}{people.csv}
\DTLloaddb{countries}{country-codes.csv}

\SetBigLabel{101mm}{139mm}{9mm}{3mm}{2mm}{2}{2}
```

3.6 Envelopes

```
\renewcommand{\PrintBigLabel}[2]{%
  \begin{minipage}[t][\LabelHeight]{\LabelWidth}%
    \vfill
    \hspace*{1em}%
    \rotatebox[origin=l]{90}{\PrintReturnAddress{#1}}\hfill
    \rule{0.1pt}{\ToAddressWidth}\space
    \rotatebox[origin=l]{90}{\PrintAddress{#2}}%
    \hspace*{1em}%
    \vfill
  \end{minipage}%
}

\makelabels

\name{Mr Big Head}
\signature{Big Head}
\location{Secret Lab of Experimental Stuff}
\address{University of Somewhere\\Some City\\AB3 4YZ}
\telephone{0123456789}

\begin{document}
```

3.6 Envelopes

```
\DTLforeach*{people}{ data
{%
 assignments
 \Id=id,%
 \Surname=surname,%
 \Forenames=forenames,%
 \Title=title,%
 \AddressI=address1,%
 \AddressII=address2,%
 \Town=town,%
 \County=county,%
 \Postcode=postcode,%
 \CountryCode=country%
}
{%
 \xDTLassignfirstmatch{countries}{code}{\CountryCode}{\CountryName=name}

\begin{letter}{\DTLifnullorempty{\Title}{}{\Title\_}%
\Forenames\_\\ \Surname\\ \AddressI\\
\DTLifnullorempty{\AddressII}{}{\AddressII\\}\\ \Town\\
\DTLifnullorempty{\County}{}{\County\\}\\ \Postcode\\ \\ \CountryName}

\opening{Dear \DTLifnullorempty{\Title}{\Forenames}{\Title} \Surname}
}
```

3.6 Envelopes

```
\thispagestyle{firstpage}
```

This is an imaginary letter.

This is the second paragraph of the letter.

```
\closing{Yours sincerely}
```

```
\ps PS: this is a postscript.
```

```
\encl{Photocopy of something interesting\\  
Photocopy of something rather dull}
```

```
\cc{Prof Important Person\\Dr Bor Ing}
```

```
\end{letter}
```

```
}
```

```
\end{document}
```

↓ Input

You can [download](#) or [view](#) this document.

3.6 Envelopes

The newlfm class automatically loads the envelab class. There's a class option `useenvelab` that's designed to activate the envelab functions but its use seems to be designed to work with newlfm's mechanism for storing addresses in an external file called `letrinfo.tex`. This doesn't fit in with the generic mail-merging functions discussed here, so instead we'll look at how to manually make address labels with envelab. This method can be applied to other classes or you can use this method if you just want to generate labels without a corresponding letter.

For the manual method, when you want to start typesetting the labels you need to use:

`\startlabels`

Definition

Then you use

`\mlabel{\langle from-address \rangle}{\langle to-address \rangle}`

Definition

for each label where `\langle from-address \rangle` is the sender's address and `\langle to-address \rangle` is the recipient's address.

EXAMPLE 24. MAIL MERGING WITH newlfm, envelab AND datatool

If you use the manual approach, you must generate the labels after you've finished typesetting the letters. This example illustrates this manual approach to generate letters and corresponding labels to everyone in the

3.6 Envelopes

sample `people.csv` file database or the `people` SQL table who has the subscribed field set:

↑ Input

```
\documentclass[stdletter,nocapaddress,avery5164biglabel]{newlfm}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[british]{babel}
\usepackage{datatool}

\newcommand{\ifcbsbool}[3]{%
  \ifboolexpr
  {
    test{\ifdefstring{#1}{true}} or
    test{\ifdefstring{#1}{1}}
  }
  {#2}{#3}%
}

\DTLloaddb{people}{people.csv}
```

3.6 Envelopes

```
\DTLloaddb{countries}{country-codes.csv}

\newlfp{orderfromtodate,sigcenter,addrfromphone,addrfromemail}

\namefrom{Mr Big Head}
\addrfrom{University of Somewhere\\Some City\\AB3 4YZ}
\emailfrom{big.head@somewhere.ac.uk}
\phonefrom{0123456789}

\regarding{A sample letter}

\closeline{Yours sincerely}

\cclist{Prof Important Person\\Dr Bor Ing}

\encllist{Photocopy of something interesting\\
Photocopy of something rather dull}

\psitem{this is a postscript}

\begin{document}
```

3.6 Envelopes

3.6 Envelopes

```
\DTLifnullorempty{\AddressII}{}{\AddressII\\}\Town\\
\DTLifnullorempty{\County}{}{\County\\}\Postcode\\CountryName
}

\greetto{Dear \DTLifnullorempty{\Title}{\Forenames}{\Title} \Surname}

\begin{newlfm}

This is an imaginary letter.

This is the second paragraph of the letter.

\end{newlfm}
}%
{)% not subscribed
}

\startlabels

\DTLforeach*{people}{% data
{%
assignments
\Id=id,%
```

3.6 Envelopes

```
\Surname=surname,%
\Forenames=forenames,%
\Title=title,%
\AddressI=address1,%
\AddressII=address2,%
\Town=town,%
\County=county,%
\Postcode=postcode,%
\CountryCode=country,%
\Subscribed=subscribed%
}
{%
\ifcsbool{\Subscribed}{%
{%
\xDTLassignfirstmatch{countries}{code}{\CountryCode}{\CountryName=name}%
\mlabel{%
{Mr Big Head\\University of Somewhere\\Some City AB3 4YZ}%
{\DTLifempty{\Title}{}{\Title\_\_}\Forenames\_\_Surname\\%
\AddressI\\%
\DTLifempty{\AddressII}{}{\AddressII\\}\Town\\%
\DTLifempty{\County}{}{\County\\}\Postcode\\}\CountryName}%
}}}
```

3.6 Envelopes

```
}%  
{% not subscribed  
}  
\end{document}
```

↓ Input

You can [download](#) or [view](#) this document.

EXERCISE 11. MAIL MERGING WITH ENVELOPE LABELS USING newlfm, envlab AND datatool

There is some duplicate code in the previous example that's inefficient, especially if you have a large database. For this exercise, rewrite the document from [Example 21](#) so that it only has one instance of each of the commands `\DTLforeach` and `\xDTLassignfirstmatch` and only one test for each member's subscribed status. Hint: recall the hook management described in [§2.1.2](#). If you're feeling adventurous try to make the address on the labels upper case without using envlab's case-changing function. (Recall `\MakeUppercase` from [Volume 2 \[96, §5.1.1\]](#).)

You can [download](#) or [view](#) the solution.

4. ■ INVOICES

There are a number of bundles for typesetting invoices on [CTAN](#) (see the [invoice topic](#) or the “[Writing Invoices](#)” of the [TeX Catalogue Topic Index \[25\]](#)) including the invoice package and the isodoc class. These are both available on MiKTeX and TeX Live and have English documentation available via [texdoc](#).

The isodoc class has already been introduced in [§3.4](#) as it can be used to create letters, so [§4.1](#) describes how to use that class to create an invoice. However, it may be that you need to add an invoice to an existing document or need to use a particular document class, so [§4.2](#) describes how to use the invoice package. Additionally, the invoice package converts currency (for foreign expenses) and computes your totals for you, whereas with isodoc you have to do the calculations yourself.

Finally, in case neither suit your requirements, [§4.3](#) describes how to create your own custom invoice using the longtable and datatool packages.

4.1 Writing an Invoice Using the *isodoc* Class

4.1 Writing an Invoice Using the *isodoc* Class

The *isodoc* class [21] can be used to create either letters (see §3.4) or invoices. To generate an invoice you need to use:

`\invoice[⟨options⟩]{⟨contents⟩}`

Definition

This is analogous to *isodoc*'s `\letter` command discussed in §3.4. As with `\letter` a `key=value` list of options can be set using the optional argument `⟨options⟩` or using the command:

`\setupdocument{⟨options⟩}`

Definition

In addition to those described in §3.4, there are also some options that relate to invoices. Some of these are described below. See the *isodoc* user guide [21] for details of the options not described in this book, which are omitted for brevity.

Some of the options that set payment information are listed below:

`term` The payment term in days.

`currency` Currency (default is euro).

`accountno` Bank account number.

*4.1 Writing an Invoice Using the *isodoc* Class*

routingno	The bank's routing number (may be omitted).
accountname	Bank account name (may be omitted).
iban	Your International Bank Account Number (enter in lower case).
bic	Your Bank Identifier Code (enter in lower case).
vatno	Your VAT number.

The options that set the payment acceptance part:

accept	Show the acceptance data.
acceptaccount	Payer's bank account number.
acceptaddress	Payer's address (separate lines with \n).
accepteuros	Euros (or equivalent) part of the amount to be paid.
acceptcents	Cents (or equivalent) part of the amount to be paid.
acceptdescription	Description.

4.1 Writing an Invoice Using the *isodoc* Class

`acceptreference` Reference.

As with `\letter` you may have multiple `\invoice` commands within a document, but the `<contents>` part is more complicated. This argument will typically contain:

`\itable{<contents>}`

Definition

This creates a two-column tabular-like environment with the given contents. You may use `&` and `\|` but the *isodoc* class provides some convenient commands to do this for you:

`\iitem{<item description>}{<amount>}`

Definition

This puts `<item description>` in the first column and `<amount>` in the second column.

`\itotal[<tag>]{<amount>}`

Definition

This adds a row for the total amount. The optional argument may be used to insert a tag, such as "Subtotal".

In addition to `\itable`, you can also use

`\accountdata`

Definition

To generate a table containing the account information needed to pay the invoice.

4.1 Writing an Invoice Using the *isodoc* Class

EXAMPLE 22. AN INVOICE (*isodoc* CLASS)

The above can be put together to create a simple invoice, listed below. Some of the options used in this example were introduced in §3.4.

↑ Input

```
\documentclass{isodoc}

\setupdocument
{
    language={en-GB},
    company={University of Somewhere},
    who={Mr Big Head},
    street={Academic Lane},
    city={Some City},
    zip={AB3 4YZ},
    country={United Kingdom},
    countrycode={GB},
    areacode={44},
    cityzip,
    date=2014-03-01,
    subject=Sample Project,
```

4.1 Writing an Invoice Using the *isodoc* Class

```
    currency={\pounds}
}

\begin{document}
\invoice
[
  ourref=1234,
  to={Miss Polly Parrot\\42 The Lane\\Some Town\\Noshire AB1 2XY}
]
{%
  \itable
  {%
    \iitem{Proof-reading}{300.00}
    \iitem{Train Fare}{43.95}
    \itotal{342.95}
  }
  \\[3ex]\accountdata
}
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 4.1](#).

4.1 Writing an Invoice Using the *isodoc* Class

University of Somewhere			
Mr Big Head Academic Lane Some City AB1 1YZ			
Miss Polly Parrot 42 The Lane Some Town Nashville AB1 2XY			
Your letter of	Your reference	Our reference	Date
		1234	1st March 2014
Subject: Sample Project			
Invoice			
Description		Amount (£)	
Postage		300.00	
Train Fare		45.95	
Total		342.95	
Banking data:			
Reference: 1234			

Figure 4.1 Invoice Using the *isodoc* Class

EXERCISE 12. CREATING AN INVOICE FOR A CUSTOMER (*isodoc* CLASS)

Create an invoice to be sent to José Arara at Nenhuma Rua, São Paulo, 123457, Brazil for 1 copy of the hardback book “Duck and Goose”: an allegory for modern times?” at 59.99, 20 copies of the paperback book “My Friend is a Duck” at 14.99 per copy, and 1 copy of the ebook “Annotated Notes on the ‘Duck and Goose’ chronicles” at 8.99. There is a promotional discount of 2.50 for this order. The cost of postage and packaging is 20.00. You will need to calculate the total for yourself.

You can [download](#) or [view](#) a solution.

FOR THE MORE ADVENTUROUS

Instead of explicitly writing the customer and order information, fetch the values either from the sample CSV files ([§2.5.1](#)) or via two join statements on the tables in the `samples` SQL database ([§2.5.3](#)). This order can be identified by the row with the “id” field equal to 2 in the sample `ordergroups.csv` file or `ordergroups` SQL table. If you use the SQL database, you can perform the summations in the SELECT statement. If you use the CSV data, then it’s more complicated. Although you can perform calculations using the commands described in [§2.1.3](#), those commands can’t be used within `\itable` as it uses the `tabularx` package [13] to layout the table which processes its contents multiple times which will throw the calculations out.

4.2 Writing an Invoice Using the *invoice* Package

Instead you will need to employ the type of method described in [Exercise 11](#) or perform two iterations over the data.

You can [download](#) or [view](#) a solution using the [CSV](#) files or [download](#) or [view](#) a solution using the [SQL](#) database.

4.2 ■ Writing an Invoice Using the *invoice* Package

At the time of writing, the current version of *invoice* is 0.9 (dated 2011-10-01). This loads the *fp* package using `\input` rather than `\usepackage`, which has unfortunate side-effects as it indirectly loads *fp.sty* through *fp.tex* but *fp.tex* is intended for Plain TeX and messes with the definition of `\ProvidesPackage` and `\RequirePackage`. Until this is fixed, you need to make sure that you load *invoice* after all your other packages.

The *invoice* package [17] defines the `invoice` environment that's used to generate the invoice.

`\begin{invoice}{\langle base\ currency\rangle}{\langle VAT\rangle}`

Definition

The first argument `\langle base\ currency\rangle` is the currency name and the second argument `\langle VAT\rangle` is the VAT percentage (without the percent sign). The `\langle VAT\rangle` may be `0`, in which case the VAT entries are hidden from the invoice,

4.2 Writing an Invoice Using the *invoice* Package

or `\$0.0`, in which case the VAT entries are displayed but show zero-rated VAT.

 Take care if you want to use a dollar sign. An error will occur if you use `\$` as the base currency unless you also load the `fontenc` package with the `T1` option. (Other options may also work, but the default `OT1` font type fails.) This problem can also happen with other currency commands, such as `\textdollar` (`textcomp` package [59]) and `\pounds`, so where possible use `fontenc` with `invoice`. If for some reason you don't want to load `fontenc`, then you can use `\string$` instead of `\$` as a workaround. However, it's a good idea in general to use the `fontenc` package anyway. If you use `fontenc` remember to use `inputenc` as well.

[FAQ: Why bother with `inputenc` and `fontenc`?]

Within the `invoice` environment you can set the project title using:

`\ProjectTitle{\langle title \rangle}`

Definition

You must have at least one project title in your `invoice` environment.

After the project title you specify the fees using:

`\Fee{\langle description \rangle}{\langle rate/unit \rangle}{\langle count \rangle}`

Definition

the local expenses using:

`\EBC{\langle description \rangle}{\langle amount \rangle}`

Definition

and the foreign expenses using:

4.2 Writing an Invoice Using the *invoice* Package

\EFC{<description>}{<foreign currency>}{<amount>}{<conversion
rate>} {<base currency result>}

Definition

You may have multiple instances of these commands. Either the fees or the expenses may be omitted, but if both are present the fees must come first.

The fees must have a description (first argument), the rate per unit of work (second argument) and the number of units (third argument). For example, a fee for proof-reading a document at a cost of £150 per day for two days:

\Fee{Proof-reading}{150}{2}

Input

The local expenses must have a description (first argument) and the amount (second argument). For example, to claim the cost of a £43.95 train ticket:

\EBC{Train fare}{43.95}

Input

The foreign expenses must have a description (first argument), the name of the foreign currency (second argument), the cost in terms of the foreign currency unit (third argument), the conversion rate (fourth argument) and the result of the currency conversion (the fifth argument). One or other of the last two arguments may be empty.

4.2 Writing an Invoice Using the *invoice* Package

For example, to charge for hotel accommodation at €300 with an exchange rate of 0.82:

```
\EFC{Hotel}{\texteuro}{300}{0.82}{}Input
```

or to charge for hotel accommodation at €300 with a local currency value of 246.67:

```
\EFC{Hotel}{\texteuro}{300}{}{246.67}Input
```

(Note that if you want to use `\texteuro` as in this example, you need to load the `textcomp` package [59].)

You can hide expenses that should contribute to the total but don't need to be itemized using:

```
\EBCi{(description)}{(amount)}Input
```

for local expenses and

```
\EFCi{(description)}{(foreign currency)}{(amount)}{(conversion rate)}  
{(base currency result)}Input
```

for foreign expenses. The arguments are the same as for `\EBC` and `\EFC`. You can make a subtotal appear for all the hidden expenses using:

4.2 Writing an Invoice Using the *invoice* Package

\STExpenses

Definition

You may also specify a discount using:

\Discount{\langle description \rangle}{\langle amount \rangle}

Definition

where *\langle description \rangle* is a description about the discount and *\langle amount \rangle* is the amount of the discount in terms of the base currency unit.

EXAMPLE 23. AN INVOICE (invoice PACKAGE)

The above can be put together to form a simple invoice:

```
\begin{invoice}{\pounds}{20}
  \ProjectTitle{Sample Project}
  \Fee{Proof-reading}{150.0}{2}
  \EBC{Train fare}{43.95}
  \EFC{Hotel}{\texteuro}{300}{0.82}{}}
\end{invoice}
```

↑ Input

↓ Input

This produces the invoice shown in Figure 4.2. You can [download](#) or [view](#) this example.

4.2 Writing an Invoice Using the *invoice* Package

Output

Sample Project			
Activity	Rate/Unit	Count	Amount (£)
Proof-reading	150.0	2	300.00
VAT (20%)			60.00
Expense			
Expense	Currency	Amount	Factor £
Train fare	£		43.95
Hotel	€	300	0.82 246.00
Sum Fees			300.00
Sum VAT			60.00
Sum Expenses			289.95
Total			649.95

Figure 4.2 Sample Invoice (*invoice* package)

4.2 Writing an Invoice Using the *invoice* Package

The invoice package provides some multilingual support so you can use it with babel [7]. If there is no support for your language, follow the instructions in the file `invoice.def` which is located in the same directory as `invoice.sty`. Alternatively, you can redefine the command names that generate the invoice tags if they don't suit your purpose. For example:

```
\renewcommand{\Fees}{Products}
\renewcommand{\UnitRate}{Price}
\renewcommand{\Count}{Quantity}
\renewcommand{\Activity}{Product}
```

↑ Input

↓ Input

EXERCISE 13. CREATING AN INVOICE FOR A CUSTOMER (invoice PACKAGE)

This exercise is like Exercise 12 except that now you need to create the document using the `invoice` package instead of the `isodoc` class (and you don't need to compute the total here as the `invoice` package does it for you).

Create an invoice to be sent to José Arara at Nenhuma Rua, São Paulo, 123457, Brazil for 1 copy of the hardback book "Duck and Goose": an allegory for modern times?" at 59.99, 20 copies of the paperback book "My Friend is a Duck" at 14.99 per copy, and 1 copy of the ebook "Annotated

4.2 Writing an Invoice Using the *invoice* Package

Notes on the ‘Duck and Goose’ chronicles” at 8.99. There is a promotional discount of 2.50 for this order. The cost of postage and packaging is 20.00.

You can choose the currency unit to suit your location, although some exchange rates might make these seem either very cheap or very expensive books, but don’t worry about that. (Remember to use `textcomp` and `fontenc` packages for the currency symbol.) In the UK, physical books are zero-rated but ebooks are subject to the standard 20% VAT rate, so for simplicity assume that the ebook price includes VAT and just use 0 to hide VAT from the invoice.

(Remember that `invoice` is a package not a class file, so you need to choose an appropriate class to use with it. For example, you might want to use it with the `letter` class so you can print it on headed paper.) You can [download](#) or [view](#) a solution.

FOR THE MORE ADVENTUROUS

Instead of explicitly writing the customer and order information, fetch the values either from the sample CSV files ([§2.5.1](#)) or via two join statements on the tables in the samples SQL database ([§2.5.3](#)). This order can be identified by the row with the “id” field equal to 2 in the sample `ordergroups.csv` file or `ordergroups` SQL table.

You can [download](#) or [view](#) a solution using the `CSV` files or [download](#) or [view](#) a solution using the `SQL` database.

4.3 Building Your Own Invoice using `longtable` and `datatool`

It may be that the layouts produced by the available classes or packages don't suit your requirements or perhaps you want to use `isodoc` but the `\itable` layout doesn't have enough columns for your needs. Since invoices typically involve aligning data in rows and columns, you can just use the `tabular` environment described in [Volume 1 \[93, §4.6\]](#) if it can fit within a single page. If the tabulated data exceeds a page, then you need to use a multi-paged tabular-like environment, such as the `longtable` environment provided by the `longtable` package [\[11\]](#).

The syntax for `longtable` is similar to `tabular`:

```
\begin{longtable}{<column specs>}
```

Definition

Unlike `tabular` the optional argument specifies the horizontal alignment instead of the vertical alignment since `longtable` isn't intended for in-line positioning. The horizontal alignment may be one of `l` (left), `c` (centre) or `r` (right). The default is `c`.

4.3 Building Your Own Invoice using *longtable* and *datatool*

The `(column specs)` are the same as for `tabular` and the rows and columns are separated in the same way using `\&` and `\&`. For example:

```
\begin{longtable}{lr}
Video & 8.99\\
CD & 9.11\\
DVD & 15.00\\
Total & 33.10
\end{longtable}
```

↑ Input

↓ Input

produces:

4.3 Building Your Own Invoice using *longtable* and *datatool*

↑ Output

Video	8.99
CD	9.11
DVD	15.00
Total	33.10

↓ Output

However, unlike `tabular`, you can also specify a caption as well as header and footer information. This is done at the start of the `longtable` environment:

```
\begin{longtable}{<column specs>}
\caption{<first page caption>}
\label{<table label>}\\
<code for first page header row>
\endfirsthead
\caption{<continuation caption>}
<code for the header row>
\endhead
```

4.3 Building Your Own Invoice using `longtable` and `datatool`

```
<code for last page footer row>
\end{lastfoot
<code for the footer row>
\end{foot
{contents}
\end{longtable}
```

You may omit any of the header, footer or caption information. The document will need at least two L^AT_EX runs to ensure the `longtable` environment correctly aligns the columns. The header and footer code will typically need to include `&` and `\\"` as per the column alignment specifications. You can use the starred version of `\caption` to suppress the numbering for example:

```
\caption*{Products (continued)}
```

Input

EXAMPLE 24. MULTI-PAGED TABULATED MATERIAL

Recall from Example 4 that datatool's `\DTLdisplaylongdb` command could be used to display the contents of a large database over multiple pages. This command internally uses the `longtable` environment. We could instead explicitly use that environment to display the data from the sample `country-codes.csv` file or `countries` SQL table:

4.3 Building Your Own Invoice using *longtable* and *datatool*

↑ Input

```
\begin{longtable}{cl}
\bfseries Country Code & \bfseries Country Name\\
\endhead
\multicolumn{2}{r}{\emph{Continued on next page}}
\endfoot
\endlastfoot
\DTLforeach*{countries}{\Code=code,\Name=name}{\Code & \Name\\}%
\end{longtable}
```

↓ Input

This sets the same header for all the pages and sets the footer to the text “Continued on next page”. Since this isn’t required for the final page, the last footer is set to empty. This produces a seven page table (without a caption). The beginning of the table looks like:

4.3 Building Your Own Invoice using *longtable* and *datatool*

Country Code	Country Name
ad	Andorra
ae	United Arab Emirates
af	Afghanistan
ag	Antigua and Barbuda
ai	Anguilla

↑ Output

The end of the first page looks like:

bw	Botswana
by	Belarus
bz	Belize
ca	Canada

↑ Output

Continued on next page

↓ Output

The final page of the table is shown in [Figure 4.3](#). You can [download](#) or [view](#) this example document.

4.3 Building Your Own Invoice using *longtable* and *datatool*

Output

Country Code	Country Name
vg	Virgin Islands, British
vi	Virgin Islands, USA
vn	Viet Nam
vu	Vanuatu
wf	Wallis and Futuna
ws	Samoa
ye	Yemen
yt	Mayotte
za	South Africa
zm	Zambia
zw	Zimbabwe

Figure 4.3 Final Page of *longtable* Displaying Countries

4.3 Building Your Own Invoice using *longtable* and *datatool*

Remember that you can use the booktabs package [24] if you want horizontal rules and you can use `\DTLiflastrow` to suppress the final `\\"` which would otherwise create unnecessary extra vertical space at the end of the table. For example:

```
\begin{longtable}{cl}
\bfseries Country Code & \bfseries Country Name\\
\midrule
\endhead
\bottomrule
\multicolumn{2}{r}{\emph{Continued on next page}}
\endfoot
\bottomrule
\endlastfoot
\DTLforeach*{countries}{\Code=code,\Name=name}%
{\Code & \Name\DTLiflastrow{}{\\"}}%
\end{longtable}
```

↑ Input

↓ Input

Alternatively you can use `\DTLiffirstrow`:

4.3 Building Your Own Invoice using *longtable* and *datatool*

↑ Input

```
\begin{longtable}{cl}
\bfseries Country Code & \bfseries Country Name\\
\midrule
\endhead
\bottomrule
\multicolumn{2}{r}{\emph{Continued on next page}}
\endfoot
\bottomrule
\endlastfoot
\DTLforeach*{countries}{\Code=code,\Name=name}%
{\DTLiffirstrow{}{\Code & \Name}\\}
\end{longtable}
```

↓ Input

As with `tabular`, the column specifiers may include `p{<width>}` for a column with multilined cells. Recall from [Volume 1 \[93\]](#), §4.6 that the `array` package [58] can be used to insert a declaration before each cell in a given column via:

4.3 Building Your Own Invoice using *longtable* and *datatool*

>{⟨declaration⟩}

Definition

directly before the column specifier. This means that if you have a column for the description of an invoiced item, you can have ragged line wrapping, which looks better than the default fully-justified paragraphs in a narrow column context.

EXAMPLE

Suppose my invoice needs four columns: the item description, the quantity ordered, the unit price and the quantity times price. The last three columns can just use the r specifier, but the first column may need a paragraph cell in the event of a long description:

↑ Input

```
\begin{longtable}{>{\raggedright}p{0.3\linewidth}rrr}
\bfseries Item & \bfseries Quantity &
\bfseries Unit Price (\pounds) &
\bfseries Price (\pounds)\\
\midrule
\endhead
``\,`Duck and Goose': an
allegory for modern times?'' (hardback) &
1 & 59.99 & 59.99\\
```

4.3 Building Your Own Invoice using *longtable* and *datatool*

```
``My Friend is a Duck'' (paperback) &  
20 & 14.99 & 299.80\\  
``Annotated Notes on the `Duck and  
Goose' Chronicles'' (ebook) &  
1 & 8.99 & 8.99\\  
``The Adventures of Duck and Goose'' (hardback) & 1 & 18.99  
& 18.99\\  
\midrule  
\multicolumn{3}{r}{\bfseries Sub-Total} & 368.78\\  
\multicolumn{3}{r}{\bfseries Postage and Packaging} & 20.00\\  
\multicolumn{3}{r}{\bfseries Promotional Discount} & -$2.50$\\  
\midrule  
\multicolumn{3}{r}{\bfseries Total} & 386.28  
\end{longtable}
```

↓ Input

This produces:

4.3 Building Your Own Invoice using *longtable* and *datatool*

↑ Output

Item	Quantity	Unit Price (£)	Price (£)
“‘Duck and Goose’: an allegory for modern times?” (hardback)	1	59.99	59.99
“My Friend is a Duck” (paperback)	20	14.99	299.80
“Annotated Notes on the ‘Duck and Goose’ Chronicles” (ebook)	1	8.99	8.99
“The Adventures of Duck and Goose” (hardback)	1	18.99	18.99
		Sub-Total	387.77
		Postage and Packaging	20.00
		Promotional Discount	-2.50
		Total	405.27

EXERCISE 14. CUSTOM INVOICE

Adapt [Exercise 12](#) so that it uses `longtable` instead of `\table`, and make it have separate columns for the quantity and unit price (as in the example above). In addition, let's now suppose the book prices exclude VAT. The physical books are zero-rated, but the ebooks are standard-rated at 20%. Add an extra column that indicates the VAT rating and add a row for the VAT after the subtotal. You can [download](#) or [view](#) a solution.

FOR THE MORE ADVENTUROUS

As with the more adventurous part of [Exercise 12](#), fetch the information from the sample CSV files or SQL database. You can [download](#) or [view](#) a solution for the CSV files or [download](#) or [view](#) a solution for the SQL data.

5. ☰ CURRICULA VITÆ (RÉSUMÉS)

There are a number of classes and packages on [CTAN](#) for typesetting a [curriculum vitae \(CV\)](#) or résumé. In fact, there's a surprisingly large list of them on the [cv topic](#) page. However, of the \LaTeX options (as opposed to Plain \TeX) some of them are old $\text{\LaTeX}2.09$ styles and some have licence issues which prevent them from being included in \TeX Live.¹ On the assumption that you're reading this chapter because you want to write a [CV](#) and are possibly pressed for time, I decided to describe only two of these options for brevity.

I discounted the non- \LaTeX options and the obsolete $\text{\LaTeX}2.09$ styles, and additionally discounted those that aren't on \TeX Live, since some readers may not want to fiddle around with manual installation. I also discounted bundles that don't have proper documentation (for example, just a README or sample file was provided). Due to my poor multilingual skills, I also discounted bundles that don't have English documentation.

¹ \TeX Live is more restrictive than MiK \TeX as it won't include any bundles that don't have an open source licence or don't provide source code for all the documentation. This means that there are some packages or classes that are available on MiK \TeX but are not on \TeX Live.

5.1 *The currvita Package*

This still left me with more than two options to choose from, so my final selection was based on how easy I found it to create a working example from the documentation. The top two were: the *currvita* package and the *europecv* class. The former, *currvita*, is simple and easy to use. The latter, *europecv*, follows the European Union [CV](#) guidelines, although the Europass title and logo can be suppressed for non-EU users. If these don't suit you, or if you are fluent in French, German or Chinese, you may prefer to investigate the other choices listed on the [cv topic](#) page.

5.1 The *currvita* Package

The *currvita* package [72] is quite simple, and should work with most classes. For example, you could just use it with the base *article* class or you might want to include your [CV](#) in a letter, in which case you might want to use one of the letter-like classes.

The available package options are as follows:

LabelsAligned Produces more compact vertical spacing.

TextAligned Produces more generous vertical spacing. This is the default vertical spacing option.

5.1 The *currvita* Package

openbib	Produces an “open” format for the bibliography.
ManyBibs	This option is provided for use with the bibunits [31] and multibib [32] packages and allows you to subdivide your publication list.
NoDate	This option suppresses the date that by default is displayed at the bottom of the CV.

The date is set using:

`\date{<date>}`

Definition

(As per `\maketitle`.) If you also want to specify a location next to the date, you can use:

`\cvplace{<location>}`

Definition

The body of the CV is contained within the cv environment:

`\begin{cv}{<heading>}`

Definition

where `<heading>` is the title text, such as “Résumé” or “Curriculum Vitae” (or “Curriculum Vitæ” if you prefer to use a ligature).

5.1 The *currvita* Package

The contents of the **CV** are typically divided into sections containing lists. These sections can be typeset within the **cv** environment using the **cvlist** environment:

`\begin{cvlist}{\langle section heading \rangle}`

Definition

where **\langle section heading \rangle** is the heading text for this list. Within the body of the **cvlist** environment, use the standard

`\item[\langle label \rangle]`

Definition

command to start each item.

EXAMPLE 25. A SAMPLE CV

The source code for this book loads the *currvita* package, so I can just use the **cv** environment within this document:

```
\date{10th March 2014}
\cvplace{My Office}
\begin{cv}{R\'esum\'e}
\begin{cvlist}{Personal Information}
\item[Name:] Polly Parrot
\item[Address:] 42 The Lane, Some Town,
```

↑ Input

5.1 The *currvita* Package

```
Noshire AB1 2XY, United Kingdom  
\item[Telephone:] 0123456789  
\item[Email:] polly.parrot@example.com  
\item[Nationality:] British  
\end{cvlist}  
\end{cv}
```

↓ Input

This produces:

Résumé

↑ Output

Personal Information

Name:	Polly Parrot
Address:	42 The Lane, Some Town, Noshire AB1 2XY, United Kingdom
Telephone:	0123456789
Email:	polly.parrot@example.com
Nationality:	British

5.1 The *currvita* Package

My Office, 10th March 2014

↓ Output

(You can [download](#) or [view](#) a complete sample document.)

Recall from [Volume 2](#) [96, §5] that you can generate a list of citations using BibTeX or biber. In a CV it's likely that you will want to include a list of publications without citing them. In this case, instead of using

`\cite{<key list>}`

Definition

you can use

`\nocite{<key list>}`

Definition

to add the citations referenced in the [comma-separated list](#) `<key list>` without producing any text. Alternatively, you can add all entries defined in your .bib file using an asterisk:

`\nocite{*}`

Input

So if you want to include a list of your publications you can use `\nocite` with `\bibliography` and `\bibliographystyle` (as described in [Volume 2](#) [96, §5.2]).

EXERCISE 15. SAMPLE CV WITH PUBLICATIONS

Modify Example 25 so that it also includes a publications list. You can either use your own .bib file or you can use the test xampl.bib file that's included in TeX distributions.

You can [download](#) or [view](#) a solution.

5.2 The europecv Class

The europecv class [112] is designed for [CVs](#) that follow the common format defined by the European Commission in 2002, but it can also be used for people outside the European Union. By default, this class will display the Europass logo and title, so you will need to include the graphicx package if you want this setting. If you're outside the European Union, you can suppress the Europass logo and title using the class options `nologo` and `notitle`:

```
\documentclass[nologo,notitle]{europecv}
```

Input

If you are in the European Union then you also need to switch to narrow Helvetica using the class options `helvetica` and `narrow`:

5.2 The *europecv* Class

↑ Input

```
\documentclass[helvetica,narrow]{europecv}  
\usepackage{graphicx}
```

↓ Input

Or if you have Arial installed:

↑ Input

```
\documentclass[arial,narrow]{europecv}  
\usepackage{graphicx}
```

↓ Input

The default paper size is US Letter, so if you want A4 paper, you need to use the a4paper option:

↑ Input

```
\documentclass[helvetica,narrow,a4paper]{europecv}  
\usepackage{graphicx}
```

↓ Input

5.2 The `europecv` Class

The body of the **CV** is placed in the `europecv` environment. So a minimal document that displays the Europass logo can be obtained with:

```
\documentclass[helvetica,narrow,a4paper]{europecv}
\usepackage{graphicx}

\begin{document}
  \begin{europecv}
  \end{europecv}
\end{document}
```

↑ Input

↓ Input

If you're not writing in English, you can set the language using the class option. This adjusts the predefined text used by `europecv` but doesn't load `babel`, so if you need `babel` (for example, you want to use the hyphenation patterns for your language) you have to load it yourself in the preamble. The `europecv` class automatically loads the `inputenc` package with the default input encoding is set to `utf8x`. If you are using a different input encoding you must set it in the class options. Remember that you also need to load the `fontenc` package.

5.2 The *europecv* Class

EXAMPLE:

If you are writing in French and using Latin 1 encoding:

```
\documentclass[helvetica,narrow,latin1,french]{europecv}  
\usepackage[T1]{fontenc}  
\usepackage{babel}
```

↑ Input

↓ Input

(*babel* will pick up the language option from the class option list.)

There are other class options as well, such as `totpages`, which will print the total number of pages on each page. See the *europecv* documentation [112] for further details.

5.2.1 Setting Personal Information

Your personal information is specified using commands that are akin to `\title` and `\author`. That is the commands just store information, so it's best to use them in the preamble.

5.2 The *europecv* Class

`\ecvname{<name>}`

Definition

Specifies your name.

`\ecvfootename{<name>}`

Definition

Specifies your name as it will appear in the footer. If omitted, it will be the same as set by `\ecvname`.

`\ecvaddress{<address>}`

Definition

Specifies your address. The address will line-wrap, but if you want to force a line break you need to use `\newline` not `\\"`.

`\ecvtelephone[<mobile>]{<telephone>}`

Definition

Sets your telephone number and optionally mobile phone number.

`\ecvfax{<fax>}`

Definition

Specifies your fax number.

`\ecvemail{<email address>}`

Definition

Specifies your email address.

5.2 The *europecv* Class

`\ecvnationality{<nationality>}` Definition

Specifies your nationality.

`\ecvdateofbirth{<date>}` Definition

Specifies your date of birth.

`\ecvgender{<gender>}` Definition

Specifies your gender.

`\ecvpicture[<options>]{<image filename>}` Definition

Specifies the name of the graphics file that contains an image of yourself. The optional argument are as used by `\includegraphics`. You can also include text before and after the image using:

`\ecvbeforepicture{<text>}` Definition

and

`\ecvafterpicture{<text>}` Definition

Within `<text>`, you may add some vertical space using:

5.2 The *europecv* Class

`\ecvspace{<height>}`

Definition

to make some minor adjustments to the picture's position. This command can't be used outside the argument of `\ecvbeforepicture` or `\ecvafterpicture`.

Once you have specified all your personal details, as described above, you need to use:

`\ecvpersonalinfo[<vspace>]`

Definition

within the `europecv` environment to display the information (analogous to `\maketitle`). The optional argument may be used to insert extra vertical space after the personal information section.

EXAMPLE 26. PERSONAL INFORMATION SECTION (*europecv* CLASS)

This example uses the `me.pdf` sample file. You can change it as appropriate.

↑ Input

```
\documentclass[helvetica,narrow,a4paper]{europecv}
```

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

```
\usepackage{graphicx}
```

```
% Specify personal data:
```

5.2 The *europecv* Class

```
\ecvname{Polly Parrot}
\ecvaddress{42 The Lane, Some Town, Noshire AB1 2XY,
United Kingdom}
\ecvtelephone[0712345678]{0123456789}
\ecvemail{polly.parrot@example.com}
\ecvnationality{British}
\ecvdateofbirth{1970-12-31}
\ecvgender{female}
\ecvpicture[width=2in]{me}% me.pdf image file

\begin{document}
\begin{europecv}
% display personal data:
\ecvpersonalinfo
\end{europecv}
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 5.1](#).

5.2 The europecv Class

The image shows a template for a EuropeCV Curriculum Vitae. At the top left is the Europass logo, which includes a stylized yellow bird icon above the word "europass". To the right of the logo, the text "Europass Curriculum Vitae" is written. Below this, there is a large, simple line drawing of a smiling person with curly hair.

Personal information

Surname(s) / First name(s)	Polly Parrot
Address(es)	The House, Some Town, Native AB1 2XY, United Kingdom
Telephone(s)	0123456789 Mobile: 0712345678
Email(s)	polly.parrot@example.com
Nationality(-ies)	British
Date of birth	1970-12-31
Gender	female

Page 1 - Curriculum Vitae
Polly Parrot

Figure 5.1 Personal Information Section (europecv class)

5.2 The *europecv* Class

5.2.2 Sections and Publication Lists

Sections in your [CV](#) are created using:

`\ecvsection[<vspace>]{<title>}`

[Definition](#)

where *<title>* is the section title and *<vspace>* is the height of the vertical space that can optionally be inserted after the title.

The text within the section is specified using:

`\ecvitem[<vspace>]{<left>}{<right>}`

[Definition](#)

where *<left>* is the text to place on the left of the vertical rule and *<right>* is the text to place on the right of the vertical rule. The optional argument is again the height of the vertical space that can be inserted after the text.

EXAMPLE 27. CURRICULUM VITÆ WITH SECTIONS (europecv CLASS)

This example adds to the code from [Example 26](#) so that it includes a section listing professional positions.

```
\documentclass[helvetica,narrow,a4paper]{europecv}  
\usepackage[T1]{fontenc}
```

 [Input](#)

5.2 The *europecv* Class

```
\usepackage{graphicx}

% Specify personal data:
\ecvname{Polly Parrot}
\ecvaddress{42 The Lane, Some Town, Noshire AB1 2XY,
United Kingdom}
\ecvtelephone[0712345678]{0123456789}
\ecvemail{polly.parrot@example.com}
\ecvnationality{British}
\ecvdateofbirth{1970-12-31}
\ecvgender{female}
\ecvpicture[width=2in]{me}% me.pdf image file

\begin{document}
\begin{europecv}
% display personal data:
\ecvpersonalinfo

\ecvsection{Professional Positions}
\ecvitem{1990--8}{Junior assistant at
``Wibblies Avian Emporium''.}
\ecvitem{1998--Present}{Senior assistant at}
```

5.2 The *europecv* Class

```
 ``The International Society of Duck and Geese  
Co-operation'' .}
```

```
\end{europecv}  
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.) The resulting document is shown in Figure 5.2.

Bibliographies are more problematic as the contents of the *europecv* environment are set using a *longtable* environment (which is why you have to use *\ecvsection* rather than *\section*). This means that you can't just use *\bibliography* within the right argument of *\ecvitem* if the bibliography is likely to span a page break. You can, however, simply place the bibliography outside the *europecv* environment, as shown below:

```
\begin{document}  
 \begin{europecv}  
 % display personal data:  
 \ecvpersonalinfo
```

↑ Input

5.2 The europecv Class

 **europass**
Curriculum Vitae

Personal information

Surname(s) / First name(s)
Adress(es)
Telephone(s)
Email(s)
Nationality(ies)
Date of birth
Gender

Professional Positions

1990-
1998-Present

Polly Parrot
The Ave, Some Town, Nothere AB1 2XY,
United Kingdom
0123456789 Mobile: 0712345678
polly.parrot@example.com
British
1970-12-31
female

Junior assistant at "Mystic Asian Emporium"
Senior assistant at "The International Society of
Duck and Geese Co-sponsor"

Page 1 - Curriculum Vitae of
Polly Parrot

Figure 5.2 Curriculum Vitæ Sections (europecv class)

5.2 The *europecv* Class

```
% start a new section:  
\ecvsection{Professional Positions}  
\ecvitem{1990--8}{Junior assistant at  
``Wibbles Avian Emporium''.}  
\ecvitem{1998--Present}{Senior assistant at  
``The International Society of Duck and Geese Co-operation''.}  
\end{europecv}  
  
\bibliographystyle{plain}  
\nocite{*}  
\bibliography{mypublications}  
\end{document}
```

↓ Input

An alternative is to use the `bibentry` package² (part of the `natbib` bundle [19]), which provides:

```
\nobibliography{(bib file)}
```

Definition

which is analogous to `\bibliography` except that it doesn't display the bibliography, and:

²The documentation for `bibentry` is inside the `bibentry.sty` file.

5.2 The *europecv* Class

\bibentry{<key>}

Definition

which displays the citation information for the reference identified by <key>.

EXAMPLE:

↑ Input

```
\documentclass[helvetica,narrow,a4paper]{europecv}
```

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

```
% add personal data here
```

```
\begin{document}
\bibliographystyle{plain}
\nocite{*}
\nobibliography{mypublications}
\begin{europecv}
% display personal data:
\ecvpersonalinfo
```

5.2 The *europecv* Class

```
% start a new section:  
\ecvsection{Professional Positions}  
\ecvitem{1990--8}{Junior assistant at  
``Wibbles Avian Emporium''.}  
\ecvitem{1998--Present}{Senior assistant at  
``The International Society of Duck and Geese Co-operation''.}  
% publications section:  
\ecvsection{Publications}  
\ecvitem{}{\bibentry{mypub1}}  
\ecvitem{}{\bibentry{mypub2}}  
\end{europecv}  
\end{document}
```

↓ Input

This assumes a file called `mypublications.bib` contains the bibliography database, including the citations with the labels `mpub1` and `mpub2`. Remember this requires a Bib \TeX run between \LaTeX runs (see [Volume 2 \[96, §5\]](#)).

Another possibility is to use the `databib` package (part of the `datatool` bundle). This has its own `.bst` Bib \TeX style that converts the bibliography data into one of `datatool`'s internal databases. This is done using:

`\DTLloaddbb[<bb>]{<db-name>}(<bib list>)`

Definition

5.2 The *europecv* Class

where $\langle bbl \rangle$ is the name of the .bbl file (defaults to `\jobname.bbl`), $\langle db-name \rangle$ is the name of the new database and $\langle bib\ list \rangle$ is the list of .bib files (without the .bib extension) where the bibliography data is stored.

As with `\bibliography` (see §5.1 and Volume 2 [96, §5.2]) you need to specify which citations you want included in the .bbl file either via:

`\cite{\langle key\ list \rangle}`

Definition

which also displays a reference in the text, or

`\nocite{\langle key\ list \rangle}`

Definition

which doesn't produce any text, but ensures that Bib_TE_X includes the references in the .bbl file.

EXAMPLE:

```
\nocite{*}
\DTLloadbbl{mypubdata}{myrefs}
```

↑ Input

↓ Input

This will create a datatool internal database called `mypubdata` that contains all the bibliographic data stored in the file `myrefs.bib`. (As with `\bibliography`,

5.2 The *europecv* Class

this requires a BibTeX run between L^AT_EX runs to ensure the citations are up-to-date.)

This database can be iterated over using `\DTLforeach` (as described in §2.7.1). However, since many of the fields will be null depending on the entry type, it's easier to iterate over the entries using:

`\DTLforeachbibentry[<condition>]{<db-name>}{<body>}`

Definition

This only makes local assignments, so it's no use in a `tabular`-like environment due to the scoping effect of `&` and `\\"`. Instead you can use:

`\gDTLforeachbibentry[<condition>]{<db-name>}{<body>}`

Definition

which makes global assignments.

As with `\DTLforeach`, there is also a starred version that performs a read-only iteration of the database. The optional argument is a conditional in the same format as the optional argument of `\DTLforeach`. There's no assignment list. Instead, you can access the citation key (as used by `\cite` and `\bibitem`) within `<body>` using:

`\DBIBcitekey`

Definition

The entry type (for example, book) is stored in:

5.2 The *europecv* Class

\DBIBentrytype

Definition

(This will always be in lower case, regardless of the case used in the .bib file.) The remaining fields can be displayed using:

\DTLbibfield{\langle field name \rangle}

Definition

or they can be assigned to a control sequence *(cs)* using:

\DTLbibfieldlet{\langle cs \rangle}{\langle field name \rangle}

Definition

In both cases, *\langle field name \rangle* is the column label, but no check is performed to determine if the column exists, so the result may be a null value (see §2.9). Available field labels are: Address, Author, BookTitle, Chapter, Edition, Editor, HowPublished, Institution, Journal, Key, Month, Note, Number, Organization, Pages, Publisher, School, Series, Title, Type, Volume, Year, ISBN, DOI, PubMed, Abstract and Url. These labels are case-sensitive (independent of the case used in the .bib file).

You can determine if a field exists within the *\langle body \rangle* part of \DTLforeachbibentry or \gDTLforeachbibentry using:

\DTLifbibfieldexists{\langle field label \rangle}{\langle true part \rangle}{\langle false part \rangle}

Definition

Since it's quite complicated working out which fields are relevant for which entry types, databib provides a convenient command that will format the entry in the current iteration according to its entry type:

5.2 The *europecv* Class

\DTLformatbibentry

Definition

By default, this only displays the fields that would typically be displayed using the standard plain bibliography style, so fields such as `Url` won't be displayed, even if they exist. This command also doesn't use `\bibitem` (recall Volume 1 [93, §5.6]). Since `\bibitem` internally uses `\item`, it's only appropriate in a list context, but it's possible it may be needed outside a list. Therefore `databib` provides:

\DTLcustombibitem{<marker code>}{<ref text>}{<key>}

Definition

This is similar to `\bibitem[<label>]{<key>}`, except that it replaces `\item` [`<label>`] with `<marker code>` and sets the cross-reference text (that is, the reference text or number generated by `\cite{key}`) to `<ref text>`. Unlike the other commands described above, `\DTLcustombibitem` may be used outside the `<body>` argument of both `\DTLforeachbibentry` and `\gDTLforeachbibentry`.

If you want to format a bibliographic entry outside of `\DTLforeachbibentry`/`\gDTLforeachbibentry` you can use:

\DTLformatthisbibentry{<db-name>}{<cite key>}

Definition

where `<db-name>` is the label identifying the database and `<cite key>` is the label identifying the reference.

EXAMPLE 28. TABULATING A BIBLIOGRAPHY

Using the above commands, it's possible to display a bibliography within a `longtable`. Recall from [Exercise 15](#) that TeX distributions come with an example file called `xampl.bib`. This has enough entries to test how well the code deals with page breaking, and so will be used in this example.

The references in this example will be numbered (rather than using an author and year system), so a counter is defined to keep track of the numbering. (See [Volume 1](#) [93, §11].)

↑ Input

```
\documentclass{article}

\usepackage{longtable}
\usepackage{databib}

\newcounter{refcount}
\newcommand*\refmark{\refstepcounter{refcount}[\therefcnt]}

\begin{document}
\nocite{*}
\DTLloaddbl{refdata}{xampl}
```

5.2 The *europecv* Class

```
\section*{Publications}

\begin{longtable}{rp{0.5\textwidth}}
\gDTLforeachbibentry{refdata}
{%
\DTLcustombibitem{\refmark}{\therefcnt}{\DBIBcitekey} &
\DTLformatbibentry\\
}%
\end{longtable}
\end{document}
```

↓ Input

This creates a four-paged document. The first page is shown in Figure 5.3. You can [download](#) or [view](#) this example document.

The data is unsorted, but remember that you can sort it using `\DTLsort` (see §2.4). For example, to sort in reverse chronological order:

```
\DTLsort*{Year=descending,Month=descending}{refdata}
```

Input

(The months should be specified using the BibTeX strings JAN, FEB etc, instead of explicitly writing the month names in order to sort the months in numerical order. You can redefine `\DTLmonthname{<number>}` to make the month names appear in a different language.)

5.2 The *europecv* Class

Publications	Output
[1] L[eslie] A. Aampert. The gnats and gnu document preparation system. <i>G-Animal's Journal</i> , 1986.	
[2] L[eslie] A. Aampert. The gnats and gnu document preparation system. <i>G-Animal's Journal</i> , 41(7):73+, July 1986. This is a full ARTICLE entry.	
[3] L[eslie] A. Aampert. The gnats and gnu document preparation system. In <i>G-Animal's Journal</i> [4], pages 73+. This is a cross-referencing ARTICLE entry.	
[4] <i>G-Animal's Journal</i> , 41(7), July 1986. The entire issue is devoted to gnats and gnu (this entry is a cross-referenced ARTICLE (journal)).	
[5] Donald E. Knuth. <i>Fundamental Algorithms</i> , chapter 1.2. Addison-Wesley, 1973.	
[6] Donald E. Knuth. <i>Fundamental Algorithms</i> , volume 1 of <i>The Art of Computer Programming</i> , section 1.2, pages 10–119. Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1973. This is a full INBOOK entry.	
[7] Donald E. Knuth. <i>Fundamental Algorithms</i> , section 1.2. Volume 1 of <i>The Art of Computer Programming</i> [11], second edition, 1973. This is a cross-referencing INBOOK entry.	
[8] Donald E. Knuth. <i>Seminumerical Algorithms</i> . Addison-Wesley, 1981.	
[9] Donald E. Knuth. <i>Seminumerical Algorithms</i> , volume 2 of <i>The Art of Computer Programming</i> . Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1981. This is a full BOOK entry.	

Figure 5.3 A Tabulated Bibliography (First Page)

5.2 The *europecv* Class

However, the database will be empty on the first run, which means you'll get an error if you try to sort it. You can test if the database is empty using:

\DTLifdbempty{\langle db-name\rangle}{\langle true part\rangle}{\langle false part\rangle}

Definition

where \langle db-name\rangle is the label identifying the database. For example:

```
\DTLifdbempty{refdata}
{%
  \DTLsort*{Year=descending,Month=descending}{refdata}%
}
```

↑ Input

↓ Input

It's therefore possible to display your list of publications in the *europecv* environment, but it will be slow (since TeX is doing the sorting and it requires iterating over a database).

5.2 The *europecv* Class

EXAMPLE 29. LIST OF PUBLICATIONS (europecv CLASS)

This example again uses the sample `xampl.bib` file that comes with \TeX distributions. You can replace `xampl` with the base name of your own personal `.bib` file.

↑ Input

```
\documentclass[helvetica,narrow,a4paper]{europecv}

\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{databib}

% add personal data here

% citation marker code:

\newcounter{refcount}
\newcommand*\refmark{\refstepcounter{refcount}[\therefcnt]}

\begin{document}
\nocite{*}
\DTLloaddbl{mypubdata}{xampl}
```

5.2 The *europecv* Class

```
\DTLifdbempty{refdata}
{}
{%
  \DTLsort*[Year=descending,Month=descending]{mypubdata}
}
\begin{europecv}
% display personal data:
\ecvpersonalinfo
% start a new section
\ecvsection{Professional Positions}
\ecvitem{1990--8}{Junior assistant at
``Wibbles Avian Emporium''}
\ecvitem{1998--Present}{Senior assistant at
``The International Society of Duck and Geese Co-operation''}
% publications section
\ecvsection{Publications}
% iterate over bib data:
\gDTLforeachbibentry*[mypubdata]{%
%
\ecvitem
{%
  left column
  \DTLcustombibitem{\refmark}{\therefcnt}{\DBIBcitekey}}%
```

5.2 The *europecv* Class

```
}%  
{\DTLformatbibentry} right column  
}%  
\end{europecv}  
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.)

You may not want to include all the citations defined in your .bib file. For example, you may only want to include your ten most recent publications or you may only want to include just your books or journal articles. This looks as though the solution can simply be obtained by filtering the rows of data, but unfortunately there's a problem: the optional argument of `\gDTLforeachbibentry` doesn't work within the `longtable` environment. Nor does `\ifthenelse` work within the final argument of `\gDTLforeachbibentry` when used inside `longtable` (although some of the etoolbox comparison commands, such as `\ifnumless` can).

When `\gDTLforeachbibentry` fails in a `tabular`-like environment, we can go back to the technique used in the solution to [Exercise 11](#) and again employed in the more adventurous section of [Exercise 12](#), where the commands described in [§2.1.2](#) were used to first build a command that could subsequently be used in the problematic environment.

EXERCISE 16. LIST OF SELECTED PUBLICATIONS (*europecv* CLASS)

Bearing in mind the above note, modify [Example 29](#) so that it just lists the ten most recent publications. You can keep track of the current row within `\DTLforeachbibentry` with the `DTLbibrow` counter. As with `\DTLforeach`, you can prematurely terminate the loop at the end of the current iteration by placing `\dtlbreak` anywhere within `<body>`.

Hint: you will need to use the standalone `\DTLformatthisbibentry` command instead of `\DTLformatbibentry` and you can check if one integer value is less than another integer value using etoolbox's

`\ifnumless{<number 1>}{{<number 2>}}{<true part>}{{<false part>}}`

[Definition](#)

If you want to compare the value of a counter, you need to use

`\value{<counter name>}`

[Definition](#)

in `<number 1>` or `<number 2>`.

You can [download](#) or [view](#) the solution to this part of the exercise.

FOR THE MORE ADVENTUROUS

Instead of just the ten most recent publications, split the publication list into the three most recent articles, the three most recent books, the two most recent conference proceedings and the two most recent booklets.

You can [download](#) or [view](#) the solution to this part of the exercise.

5.2.3 Spoken Languages

The `europecv` class also has commands to produce the section on your spoken language skills. This section is started by identifying your mother tongue using:

`\ecvmothertongue[<vspace>]{<language>}`

Definition

For example:

`\ecvmothertongue{English}`

Input

The language table header is typeset using:

`\ecvlanguageheader{<symbol>}`

Definition

where `<symbol>` is a footnote symbol used in the table footer. Each row of the language table is then typeset using:

`\ecvlanguage[<vspace>]{<language>}{<l1>}{<l2>}{<l3>}{<l4>}{<l5>}`

Definition

except for the last, which is typeset using:

5.2 The *europecv* Class

`\ecvlastlanguage[<vspace>]{<language>}{<l1>}{<l2>}{<l3>}{<l4>}{<l5>}` Definition

where $\langle vspace \rangle$ again indicates any vertical space that should be inserted after the row, and $\langle language \rangle$ is the name of the language. The other five arguments $\langle l1 \rangle, \dots, \langle l5 \rangle$ should be brief descriptions relating to:

$\langle l1 \rangle$ understanding (listening);

$\langle l2 \rangle$ understanding (reading);

$\langle l3 \rangle$ speaking (spoken interaction);

$\langle l4 \rangle$ speaking (spoken production);

$\langle l5 \rangle$ writing.

Each of these arguments should be in the form:

`\ecvCEF{<level>}{<descr>}` Definition

where $\langle level \rangle$ is the self-assessment level code and $\langle descr \rangle$ is a brief description. There are some convenient shortcuts described in [Table 5.1](#).

After the last language row, the table footer is typeset using:

`\ecvlanguagefooter[<vspace>]{<symbol>}` Definition

where $\langle symbol \rangle$ should be the same as used in `\ecvlanguageheader`.

5.2 The *europecv* Class

Table 5.1 Convenient Shortcut Commands for \ecvCEF

Shortcut	Expansion
\ecvAOne	\ecvCEF{A1}{basic user}
\ecvATwo	\ecvCEF{A2}{basic user}
\ecvBOne	\ecvCEF{B1}{independent user}
\ecvBTwo	\ecvCEF{B2}{independent user}
\ecvCOne	\ecvCEF{C1}{proficient user}
\ecvCTwo	\ecvCEF{C2}{proficient user}

EXAMPLE

```
\ecvmothertongue[10pt]{English}
\ecvlanguageheader{(*)}
\ecvlanguage{French}{\ecvCOne}{\ecvCTwo}{\ecvBTwo}{\ecvCOne}
{\ecvCTwo}
\ecvlastlanguage{German}{\ecvATwo}{\ecvATwo}{\ecvATwo}{\ecvATwo}
{\ecvATwo}
\ecvlanguagefooter{(*)}
```

↑ Input

↓ Input

5.2 The *europecv* Class

⚠ Unless you use either narrow Arial or condensed Helvetica, the language table may appear cramped. For this example to work, I used the following settings to adjust the left column width and the gap on either side of the vertical rule:

```
\ecvLeftColumnWidth{2cm}  
\ecvColSep{4pt}
```

↑ Input

↓ Input

and set the default font size to 10pt:

```
\documentclass[helvetica,narrow,a4paper,10pt]{europecv}
```

Input

This produces the output shown in [Figure 5.4](#) unless you use the `booktabs` class option:

```
\documentclass[helvetica,narrow,a4paper,10pt,booktabs]{europecv}
```

Input

in which case it produces the output shown in [Figure 5.5](#).

5.2 The *europecv* Class

Mother tongue(s)	English				Output
Self-assessment European level ^(*)	Understanding		Speaking		Writing
	Listening	Reading	Spoken interaction	Spoken production	
	C1 Proficient user	C2 Proficient user	B2 Independent user	C1 Proficient user	C2 Proficient user
French	A2 Basic user	A2 Basic user	A2 Basic user	A2 Basic user	A2 Basic user
German					

^(*)Common European Framework of Reference (CEF) level

Figure 5.4 Example Language Table

5.2 The *europecv* Class

Mother tongue(s)		English						Output
Self-assessment	European level ^(*)	Understanding		Speaking		Writing		
		Listening	Reading	Spoken interaction	Spoken production			
French	C1 Proficient user	C2 Proficient user	B2 Independent user	C1 Proficient user	C2 Proficient user			
German	A2 Basic user	A2 Basic user	A2 Basic user	A2 Basic user	A2 Basic user			

^(*)Common European Framework of Reference (CEF) level

Figure 5.5 Example Language Table (booktabs option)

6. OFFICIAL DOCUMENTS

This chapter covers topics related to official documents, including records (such as minutes or agendas of meetings), memos and press releases. The chapter also covers typesetting matters relating to confidential documents and legal documents, such as watermarks and redaction.

6.1 Memos

[§3.3](#) introduced the `newlfm` class [107], which can be used for writing correspondence. This class can also be used for writing memos and press releases. (See the [next section](#) for press releases.) If you read the earlier section on the `newlfm` class, you may remember that you can set the style of the document through the class options. For example, the `stdletter` option sets the standard letter style. There are two styles relating to memos: `stdmemo` (standard memo) and `fullmemo` (full memo).

As before, you still use the `newlfm` environment. However, some of the commands described in [§3.3](#) are ignored for the memo styles, such as

6.1 Memos

those that set the greeting, closing and signature. Again, options can be set using either the class option list or the command:

`\newlfmP{<options list>}`

Definition

Memo options are listed in [Table 6.1](#).

Don't be confused by the "Re:" line. The `\regarding` command described in [§3.3](#) is for letter subject lines. (Prefixed with "Regarding:") In a memo, the text in the line prefixed with "Re:" is set using:

`\re{<text>}`

Definition

 Note that in the current version of newlfm (dated 2009-04-10) there are bugs in the `memonofrom`, `memonoto` and `memonore` options. The following lines are workarounds to implement these options:

```
\setboolean{@memo@e}{false}% memonofrom  
\setboolean{@memo@g}{false}% memonoto  
\setboolean{@memo@f}{false}% memonore
```

↑ Input

↓ Input

6.1 Memos

Table 6.1 Memo Options

Option	Description
<code>memonofrom</code>	Omit sender's block.
<code>memoemailfrom</code>	Include sender's email.
<code>memoaddrfrom</code>	Include recipient's address.
<code>memophonefrom</code>	Include sender's telephone number.
<code>memofaxfrom</code>	Include recipient's fax number.
<code>memopagerfrom</code>	Include recipient's pager number.
<code>memonoto</code>	Omit recipient's block.
<code>memoemailto</code>	Include recipient's email.
<code>memoaddrto</code>	Include recipient's address.
<code>memophoneto</code>	Include recipient's telephone number.
<code>memofaxto</code>	Include recipient's fax number.
<code>memopagerto</code>	Include recipient's pager number.
<code>memodate</code>	Set the date on the memo.
<code>fullmemo</code>	Use all optional items.
<code>memonore</code>	Omit the "Re:" line.

6.1 Memos

EXAMPLE 30. SAMPLE MEMO (newlfm CLASS)

Commands such as `\nameto` and `\dateset` were described earlier in §3.3. This example makes use of some of those commands, but sets the style to `stdmemo` to create a memo instead of a letter.

↑ Input

```
\documentclass[12pt]{newlfm}

\usepackage[british]{babel}

\newlfmP{stdmemo,memoemailfrom,memophonefrom,memoaddrfrom,memodate}

\dateset{11-03-2014}
\nameto{Mabel Canary}

\namefrom{Mr Big Head}
\emailfrom{big.head@somewhere.ac.uk}
\phonefrom{0123456789}
\addrfrom{Secret Lab of Experimental Stuff}

\re{Cricket Match}
```

6.2 Press Releases

```
\begin{newlfm}
```

It has come to my attention that certain members of your team intend to bring the prototype ray gun to the forthcoming cricket match against the Department of Stripy Confectioners in order to 'level the playing field'. Please remind them that this is against regulations. Also, the mind-controlling cookies are inappropriate for the tea interval provisions.

```
\end{newlfm}
```

```
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in Figure 6.1.

6.2 Press Releases

The newlfm class can produce a press release using the `pressrelease` style. There are only two press release options: `dspace` (double-spaced, default) and `sspace` (single-spaced). This style has wide margins, a plain format and ends with three hashes (# # #). This is fairly standard for press releases, so don't try jazzing it up. As with the memo styles, some of the commands

6.2 Press Releases



Figure 6.1 A Sample Memo (`newlfm` class)

6.2 Press Releases

described in §3.3 are ignored for the press release style, but some of the commands used by the memo styles are also ignored.

The subject of the press release is set using:

`\headline{<text>}`

Definition

By default, the phrase “For Immediate Release” is placed at the start. This wording can be changed using

`\release{<text>}`

Definition

EXAMPLE 31. PRESS RELEASE (newlfm CLASS)

Note that the `\emailfrom` command is ignored by the `pressrelease` style, so it hasn’t been included. The `\headline` command has been used to set the title and there is also a `\section` command available that produces an unnumbered section. The `\url` command is provided by the `url` package [5].

```
\documentclass[newlfm]{  
    \usepackage[url]  
    \usepackage[british]{babel}}
```

↑ Input

6.2 Press Releases

```
\newlfmP{pressrelease}

\namefrom{Mr Big Head}
\addrfrom{University of Somewhere\\Some City\\AB3 4YZ}
\phonefrom{0123456789}
```

```
\headline{Secret Experimental Lab Open Day}
```

```
\begin{document}
```

```
\begin{newlfm}
```

The University of Somewhere is throwing open the doors
of its renowned Secret Experimental Lab to the public
for the first time ever on 1st April 2014.

Mr Big Head, managing director of the lab, will be giving
conducted tours between 11:59 and 12:00. More details,
including car parking arrangements, are listed at

```
\url{www.somewhere.ac.uk/secretopenday}
```

```
\section{About the University of Somewhere}
```

6.2 Press Releases

The university is a non-existent academic institution renowned for its cutting-edge imaginary research.

```
Website: \url{www.somewhere.ac.uk}  
\end{newlfm}
```

```
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this example.) The resulting document is shown in [Figure 6.2](#).

The newlfm class is quite difficult to adapt and it seems strange not to have an email option for a press release in this Internet age. Since this didn't suit my purposes and I couldn't find another class for press releases, I decided to write my own class called `pressrelease` [97], which I uploaded to [CTAN](#). At the time of writing, the current version is 1.0 (dated 2014-09-10).

The `pressrelease` class has the following class options:

`10pt` Set the normal font size to 10pt.

`11pt` Set the normal font size to 11pt.

6.2 Press Releases

For Immediate Release

Contact: Mr Big Head
University of Somewhere
Somewhere
AB3 4YZ
Telephone: 0123456789
Date: 10th September 2014

Secret Experimental Lab Open Day

The University of Somewhere is throwing open the doors of its renowned Secret Experimental Lab to the public for the first time ever on 1st April 2014.

Mr Big Head, managing director of the lab, will be giving conducted tours between 11:00 and 12:00. More details, including car parking arrangements, are listed at www.somewhere.ac.uk/secretopenday

About the University of Somewhere

The university is a non-existent academic institution renowned for its cutting-edge imaginary research.

Website: www.somewhere.ac.uk

#

1 of 1

Figure 6.2 A Sample Press Release (newlfm class)

6.2 Press Releases

`12pt` Set the normal font size to 12pt.

`a4paper` Set the paper size to A4.

`letterpaper` Set the paper size to US Letter.

`symbols` Use symbols instead of textual tags.

As with newlfm there are commands to specify the contact information.

The main headline text is specified using:

`\PRheadline{<text>}`

Definition

and the sub-heading (if required) is specified using:

`\PRsubheadline{<text>}`

Definition

The date of the press release is specified using the standard `\date` command.

The company logo (if required) is specified using:

`\PRlogo{<logo>}`

Definition

where `<logo>` is the command or commands required to input or draw the logo. For example, if the logo is in the image file `company-logo.png`:

6.2 Press Releases

\PRlogo{\includegraphics{company-logo}}

Input

(Remember that you need to load the graphicx package if you want to use the \includegraphics command.)

The company's name is specified using:

\PRcompany{\langle name\rangle}

Definition

The department's name is specified using:

\PRdepartment{\langle name\rangle}

Definition

The company's location is specified using:

\PRlocation{\langle location\rangle}

Definition

The company's address is specified using:

\PRaddress{\langle address\rangle}

Definition

(You may use \\ to separate the lines in the address.)

The company's email address is specified using:

\PRemail{\langle address\rangle}

Definition

The company's telephone number is specified using:

6.2 Press Releases

`\PRphone{<number>}`

[Definition](#)

The company's contact mobile number is specified using:

`\PRmobile{<number>}`

[Definition](#)

The company's fax number is specified using:

`\PRfax{<number>}`

[Definition](#)

The company's website is specified using:

`\PRurl{<website>}`

[Definition](#)

The company's opening hours are specified using:

`\PRhours{<times>}`

[Definition](#)

For example:

`\PRhours{Mon--Fri 9:00--17:00}`

[Input](#)

As with titling commands, such as `\title`, the above commands just store the information and may be placed in the preamble. The actual contents of the press release should go in the `pressrelease` environment. Within this environment, you can use the `about` environment to add information about the company.

6.3 Minutes

```
\begin{pressrelease}  
<press release text>
```

```
\begin{about}  
(information about the company)  
\end{about}  
\end{pressrelease}
```

Definition

EXERCISE 17. PRESS RELEASE (pressrelease CLASS)

Adapt Example 31 so that it uses the pressrelease class and add an email address to the contact details. You can [download](#) or [view](#) a solution.

6.3 Minutes

There are only three options for writing minutes listed on the [meeting-admin topic](#) page: the meetingmins class (which can also typeset agendas but has no multilingual support), the minutes package and the protocol class. The last one, protocol, only has German documentation, so it's not discussed here.

6.3 Minutes

6.3.1 The meetingmins Class

The `meetingmins` class [6] uses the standard sectioning commands, such as `\section` and `\subsection`, but provides additional commands and environments useful in the creation of minutes or agendas. Note that this class doesn't provide multilingual support.

The default behaviour of this class is to assume you are creating minutes from a meeting. The document will be headed "Minutes for `\langle date \rangle`" and some of the commands described below will be ignored. You can change this format via one of the following class options:

`agenda` This indicates the document is an agenda. The heading will be "Agenda for `\langle date \rangle`".

`chair` This indicates the document is the chair's agenda. The heading will be "Chair's Agenda for `\langle date \rangle`" and there will be a list of members names with checkboxes for the chair to track attendance.

`notes` This sets the heading to "Notes for `\langle date \rangle`".

The date of the meeting is set using:

6.3 Minutes

`\setdate{<date>}`

Definition

The name of the committee is set using:

`\setcommittee{<name>}`

Definition

The list of members is set using:

`\setmembers{<list>}`

Definition

where *<list>* is a list of members. (The member list is only used by the chair class option.) The list of people present at the meeting is set using:

`\setpresent{<list>}`

Definition

where again *<list>* is a list of names. (This list is ignored by the chair and agenda options.) Within *<list>*, for both `\setmembers` and `\setpresent`, the chair's name should be set with:

`\chair{<name>}`

Definition

EXAMPLE:

```
\setdate{12th March 2014}
```

↑ Input

6.3 Minutes

```
\setcommittee{Secret Lab of Experimental Stuff}
\setmembers{
    \chair{Mr Big Head},
    Prof Important Person,
    Dr Bor Ing
}
\setpresent{
    \chair{Mr Big Head},
    Dr Bor Ing
}
```

↓ Input

This sets information (analogous to `\author` and `\title`) but doesn't actually display any information, so these commands may be used in the preamble. As with `\author` and `\title`, you need to use

`\maketitle`

Definition

to make the title information appear.

Within the `document` environment, if you need numbered lists you can use the `items` environment

6.3 Minutes

```
\begin{items}
\item <item text>
...
\end{items}
```

Definition

within sections and the `subitems` environment

```
\begin{subitems}
\item <item text>
...
\end{subitems}
```

Definition

within subsections and sub-subsections. As is usual for list environments, use `\item` at the start of each item. If there are items that should only appear in the chair's agenda or in the notes, you can instead use the `hiddenitems` environment

```
\begin{hiddenitems}
\item <item text>
...
\end{hiddenitems}
```

Definition

within sections and the `hiddensubitems` environment

6.3 Minutes

```
\begin{hiddensubitems}
\item <item text>
...
\end{hiddensubitems}
```

Definition

within subsections and sub-subsections. There is also a similar `hiddentext` environment

```
\begin{hiddentext}
<text>
\end{hiddentext}
```

Definition

for non-list text. The contents `<text>` is only displayed if the `chair` or `notes` class option has been used.

The date and time of the next meeting can be displayed using:

```
\nextmeeting{<date and time>}
```

Definition

This is ignored for the `agenda` option. Finally

```
\priormins
```

Definition

is a convenient shortcut for “The minutes of the previous meeting were approved”.

EXAMPLE:

6.3 Minutes

↑ Input

```
\section{Announcements}
\begin{hiddenitems}
\item A~fire alarm drill is expected during the meeting.
\end{hiddenitems}

\section{Old Business}
\begin{items}
\item \priormins
\end{items}

\section{New Business}
\begin{items}
\item Discuss schedules for secret research.

\item Discuss schedules for new experimental stuff.
\end{items}

\nextmeeting{15th April 2014 at 15:00}
```

↓ Input

6.3 Minutes

If the chair or notes class options are used the fire alarm announcement will be displayed, otherwise it will be skipped.

EXERCISE 18. MINUTES AND AGENDAS (meetingmins CLASS)

Create a document using the meetingmins class that uses all the commands described in this section. Try using the different class options to see how they affect the document. (Remember that you can't mix the class options.)

You can [download](#) or [view](#) a solution.

6.3.2 The minutes Package

Unlike meetingmins, described [above](#), minutes [53] is a package, so you can choose your document class. The package comes with alternative German commands that you can use instead of the English command names. For example, instead of using `\subtitle` you can use `\untertitel`. For brevity, this book only describes the English commands. See the minutes documentation for a list of the German equivalents and for the additional Dutch support. The default language is German, so if you are writing the minutes in English, you'll need to load `babel` [7] with the `english` option.

The meeting minutes are contained in the body of the `Minutes` environment

6.3 Minutes

\begin{Minutes}{\langle title\rangle}

Definition

where $\langle title \rangle$ is the title of the minutes. You can have more than one `Minutes` environment within your document, for example, if you want a compilation of all the minutes for a particular group or committee.

Within the `Minutes` environment, you can set various information using the commands described below.

\subtitle{\langle title\rangle}

Definition

This sets the subtitle, if required.

\moderation{\langle name\rangle}

Definition

This sets the name of the meeting moderator (for example, the chair).

\minutetaker{\langle name\rangle}

Definition

This sets the name of the minute taker.

\participant{\langle names\rangle}

Definition

This sets the names of the people present at the meeting.

6.3 Minutes

`\guest{\langle names\rangle}`

Definition

This sets the names of any guests present at the meeting.

`\minutesdate{\langle date\rangle}`

Definition

This sets the date of the meeting.

`\starttime{\langle time\rangle}`

Definition

This sets the starting time of the meeting.

`\endtime{\langle time\rangle}`

Definition

This sets the time the meeting ended.

`\location{\langle place\rangle}`

Definition

This sets the location of the meeting.

`\cc{\langle names\rangle}`

Definition

This sets the distribution list. The argument `\langle names\rangle` is a list of names of people who should receive a copy of the minutes. To specify absentees, you can either use

6.3 Minutes

\missingExcused{\langle excused names\rangle}

Definition

and

\missingNoExcuse{\langle no-excuse names\rangle}

Definition

or

\missing[{\langle excused names\rangle}]{\langle no-excuse names\rangle}

Definition

where \langle excused names\rangle is a list of names of missing people who provided an excuse and \langle no-excuse names\rangle is a list of missing people who didn't provide an excuse.

The above commands all behave in an analogous way to \title and \author. Once they have been specified, you then need to use:

\maketitle

Definition

EXAMPLE 32. SAMPLE MINUTES (minutes PACKAGE)

This example just sets up the title information for the minutes.

```
\documentclass{article}
```

↑ Input

6.3 Minutes

```
\usepackage[english]{babel}
\usepackage{minutes}

\begin{document}

\begin{Minutes}{Secret Lab of Experimental Stuff}
  \subtitle{Annual General Meeting}
  \moderation{Mr Big Head}
  \minutetaker{Dr Bor Ing}
  \participant{Polly Parrot, Mabel Canary}
  \missing[Z\"oe Zebra, Jos\'e Arara]{Dickie Duck, Fred Canary}
  \guest{Prof Important Person}
  \minutesdate{12th March 2014}
  \starttime{15:00}
  \endtime{17:00}
  \location{University of Somewhere}
  \cc{Vice Chancellor}

  \maketitle
\end{Minutes}

\end{document}
```

↓ Input

6.3 Minutes

The result is shown in [Figure 6.3](#). Since this document only contains the minutes from a single meeting, I haven't bothered to include an overall document title or table of contents. This causes warnings from the minitoc package [22] (which the minutes package loads). If you want to have a collection of minutes, you can add the title and contents at the start of the document:

```
\begin{document}
\title{Minutes from the Secret Lab Meetings}
\author{Dr Bor Ing}
\maketitle
\tableofcontents
% add the Minutes environments here
```

↑ Input

↓ Input

You can [download](#) or [view](#) this example.

The minutes can be subdivided into topics using:

Secret Lab of Experimental Stuff

Annual General Meeting

Moderation Mr Big Head

Minutes taker Dr Bor Ing

Those present Polly Parrot, Mabel Canary

Absent (excused) Zöe Zebra, José Arara

Absent (not excused) Dickie Duck, Fred Canary

Guest Prof Important Person

Location of the meeting University of Somewhere

Date 12th March 2014 15:00–17:00

Distribution Vice Chancellor

Figure 6.3 Sample Minutes (Title Information Only)

6.3 Minutes

\topic[⟨toc text⟩]{⟨text⟩}

Definition

and subtopics using:

\subtopic[⟨toc text⟩]{⟨text⟩}

Definition

where ⟨text⟩ is the topic or subtopic and ⟨toc text⟩ is alternative text for the table of contents. These commands are analogous to sectioning commands such as \section. The ⟨toc text⟩ (or ⟨text⟩ if the optional argument is omitted) appears in an overview section in the minutes and also appears in the overall document table of contents (if \tableofcontents has been used, as described above).

Tasks can be specified using the \task command which has a starred and unstarred version. The unstarred version has the syntax:

\task[⟨footnote text⟩]{⟨name⟩}{⟨when⟩}{⟨text⟩}

Definition

The starred version has the syntax:

\task*[⟨when⟩]{⟨text⟩}

Definition

EXAMPLE:

↑ Input

6.3 Minutes

```
\topic{Tasks}
\subtopic{New Experimental Stuff}
\task[done]{Mabel Canary}[tomorrow]{Proposal for a time machine}
\task[pending]{Polly Parrot}{Apply for a ray gun grant}

\subtopic{Kitchen}
\task*[Order a new coffee machine]
\task*[today]{Remove the mind-controlling cookies}
```

↓ Input

As with the meetingmins class, the minutes package allows you to hide text. This is done either via the command:

```
\secret{(secret text)}
```

Definition

or using the `Secret` environment

```
\begin{Secret}
<secret text>
\end{Secret}
```

Definition

The `<secret text>` will only be displayed if you use the `Secret` package option.

```
\usepackage[Secret]{minutes}
```

Input

6.3 Minutes

EXAMPLE:

```
\begin{Secret}
\task*{There will be a surprise lab inspection on Tuesday.}
\end{Secret}
```

↑ Input

↓ Input

If the meeting discussed an opinion, this can be recorded using:

```
\opinion{<main>}{|<differing>}
```

Definition

where *<main>* is the main opinion held and *<differing>* is the differing opinion. The discussion can then be formatted using the [Opinions](#) environment:

```
\begin{Opinions}
\item[<name>] <opinion>
...
\end{Opinions}
```

Definition

EXAMPLE:

6.3 Minutes

↑ Input

```
\opinion
{Keep the coffee break at 11.00am}%
{Move the coffee break to 10:30am}%
\begin{Opinions}
\item[Mabel Canary] We should continue to have
coffee at 11:00am.
\item[Polly Parrot] We should move the coffee
break to an earlier time.
\end{Opinions}
```

↓ Input

Arguments can be formatted using the `Argumentation` environment.

```
\begin{Argumentation}
⟨items⟩
\end{Argumentation}
```

Definition

Within this environment, you can use the standard `\item` command for a comment or one of the following commands:

6.3 Minutes

\pro *<reason for>*

Definition

which itemizes a reason in favour of the argument,

\Pro *<important reason for>*

Definition

which itemizes an important reason in favour of the argument,

\contra *<reason against>*

Definition

which itemizes a reason against the argument,

\Contra *<important reason against>*

Definition

which itemizes an important reason against the argument, and

\result *<argument result>*

Definition

which itemizes the result of the argument.

EXAMPLE:

```
\begin{Argumentation}
```

↑ Input

```
    \pro We've always had coffee at 11:00am. There's no need to  
    change it.
```

6.3 Minutes

```
\contra 11:00am is too long a wait for the caffeine addicts.  
\Pro Coffee at 10:30am would interfere with our clandestine  
experiments scheduled at that time.  
\item Prof Important Person said it would be better to have  
tea instead of coffee.  
\result The coffee break will continue to be at 11:00am.  
\end{Argumentation}
```

↓ Input

A single vote can be formatted using

```
\vote{<description>}{'yes'}{'no'}{'abstain'}[<decision>]
```

Definition

where *<description>* is a brief description of the vote, *<yes>* is the number of “Yes” votes, *<no>* is the number of “No” votes and *<abstain>* is the number of abstainers. Optionally, a decision can be added. For example:

```
\vote{Maintain coffee at 11am}{2}{1}{1}
```

Input

indicates that there were two votes in favour of maintaining coffee at 11am, one vote against and one abstainer.

Multiple votes can be listed in the **Vote** environment:

6.3 Minutes

```
\begin{Vote}
\vote{(description)}{(yes)}{(no)}{(abstain)}[(decision)]
...
\end{Vote}
```

Definition

EXAMPLE:

```
\begin{Vote}
\vote{Maintain coffee at 11am?}{2}{1}{1}
\vote{Move clandestine experiments to after lunch?}{1}{3}{0}
\end{Vote}
```

↑ Input

↓ Input

Decisions are first declared using:

```
\decisiontheme{(theme)}{(title)}
```

Definition

This doesn't display anything in the document at this point, but it will be added to the list of decisions which can be displayed using:

```
\listofdecisions
```

Definition

This is like other `\listof...` commands, such as `\listoftables`. Each decision is then specified using:

6.3 Minutes

\decision{<theme>}{<short description>} [<long description>]

Definition

There is also a starred version which doesn't add the decision to the list of decisions:

\decision*{<short description>} [<long description>]

Definition

EXAMPLE:

```
\decision{Ray Guns}{Should we reverse the polarity of  
ray guns?}  
\decision{Ray Guns}{The ray gun polarity doesn't need  
modifying.}  
\decision*{We don't need to reverse the polarity.}  
  [Reversing the polarity is generally considered to be a daft  
idea.]
```

↑ Input

↓ Input

If necessary, the minutes can be signed at the end using

6.3 Minutes

`\signature{\langle name\rangle}`

Definition

This should be placed before the end of the `Minutes` environment. Any additional information that doesn't belong to the minutes may be included in the `Postscript` environment

`\begin{Postscript}`
(additional information)
`\end{Postscript}`

Definition

or in the argument of

`\postscript{\langle additional information\rangle}`

Definition

For other commands not listed here, including how to alter the style, see the minutes documentation.

EXERCISE 49. MINUTES (minutes PACKAGE)

Extend the document in [Example 32](#) to include topics, tasks, opinions, arguments, votes and decisions.

You can [download](#) or [view](#) a solution.

6.4 Confidentiality

This section covers some topics related to confidentiality. There are a number of options listed on the [security topic](#) page. This section will just be looking at redaction ([§6.4.1](#)) and watermarks ([§6.4.2](#)). See also [§2.3](#) for advice on document security.

6.4.1 Redaction: The censor Package

The `censor` package [87] provides a way to black out redacted words or phrases, paragraphs, or boxes (such as included graphics or a `tabular` environment). Note that the code used for redaction affects line-breaking and paragraph justification, so don't expect the redacted version of the document to look as well as an unedited version.

Redacted words or phrases are tagged using:

`\censor{(text)}`

Definition

EXAMPLE:

 Input

6.4 Confidentiality

Following the research group's unsuccessful attempt to create \censor{mind-controlling} \censor{cookies}, they will now be working on a new \censor{ray gun}.

↓ Input

produces:

Following the research group's unsuccessful attempt to create [REDACTED]
[REDACTED], they will now be working on a new [REDACTED]

↑ Output

↓ Output

Note that the argument of \censor is placed in a box so its contents can't be broken across a line, which is why I split up "mind-controlling" and "cookies". In fact, it may even be necessary to do:

\censor{mind}-\censor{controlling} \censor{cookies},

Input

A box can be redacted using:

\censorbox[⟨declarations⟩]{⟨box⟩}

Definition

where ⟨box⟩ is the box being redacted and ⟨declarations⟩ are any commands (such as font-changing declarations) that should be applied before the start of the box.

6.4 Confidentiality

EXAMPLE:

```
\begin{table}
  \caption{A Redacted Table}
  \label{tab:redacted}
  \centering
  \censorbox{%
    \begin{tabular}{lc}
      \bfseries Project & \bfseries Success Rate \\
      Mind-controlling Cookies & 2\% \\
      Telepathic Cakes & 1\% \\
      Exploding Chocolates & 25\%
    \end{tabular}%
  }
\end{table}
```

↑ Input

↓ Input

This produces [Table 6.2](#).

A paragraph can be redacted using:

```
\blackout{<text>}
```

Definition

6.4 Confidentiality

Output

Table 6.2 A Redacted Table



EXAMPLE:

↑ Input

\blackout{Following the research group's unsuccessful attempt to
create mind-controlling cookies, they will now be working on
a new ray gun.}

↓ Input

produces:

↑ Output

[REDACTED]

↓ Output

6.4 Confidentiality

Note that there are restrictions on the use of `\blackout`: $\langle text \rangle$ can't end with "glue", such as a space or `EOL` character; periods aren't redacted; it can't be used across changes in scope, such as environment boundaries or across cells within a `tabular`-like environment. This command also shows spaces between words (unless the spaces are hidden within a `token`), but there is an alternative command that hides these spaces:

`\xblackout{\langle text \rangle}`

Definition

EXAMPLE:

```
\xblackout{Following the research group's unsuccessful attempt to  
create mind-controlling cookies, they will now be working on  
a new ray gun.}
```

↑ Input

↓ Input

produces:

```
[REDACTED]
```

↑ Output

↓ Output

6.4 Confidentiality

This command also suffers from drawbacks. For example, the redaction can cause lines to protrude into the left and right margins. See the `censor` documentation for further details.

The unredacted version of the document can be created by placing

`\StopCensoring`

Definition

at the beginning of the document. You can also restart redaction with:

`\RestartCensoring`

Definition

It's possible that you may need to work on your document at an insecure location. In which case, you won't want your sensitive information in your `.tex` file. The `censor` package provides:

`\censor*{<size>}`

Definition

where `<size>` is the approximate width (in ex) of the redacted text, and:

`\censorbox*[{<declarations>}]{<width>}{<height>}{<depth>}`

Definition

where `<width>` is the approximate width (in ex) of the box, `<height>` is the approximate height (in multiples of `\baselineskip`) and `<depth>` is the approximate depth (in multiples of `\baselineskip`). The optional argument should be used to specify any commands (such as font changing declara-

6.4 Confidentiality

tions) that may effect the size of an ex or the value of `\baselineskip`. (For example, `\small`.) There's no starred version of `\blackout` or `\xblackout`.

EXAMPLE:

↑ Input

```
\newcommand*{\OldProject}{\censor*{4}-\censor*{11} \censor*{7}}
\newcommand*{\NewProject}{\censor*{3} \censor*{3}}
\newcommand*{\SuccessRates}{\censorbox*{40}{4}{0}}
```

Following the research group's unsuccessful attempt to create `\OldProject`, they will now be working on a new `\NewProject`. The success rate of previous projects is shown in Table~`\ref{tab:success}`.

```
\begin{table}
  \caption{Project Success Rates}
  \label{tab:success}
  \centering
  \SuccessRates
\end{table}
```

↓ Input

6.4 Confidentiality

If the censoring is on, the redacted text will again be replaced by filled rectangles:

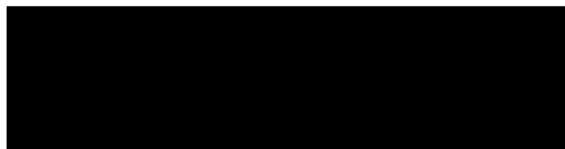
Following the research group's unsuccessful attempt to create █-█
█, they will now be working on a new █ █. The success rate of previous projects is shown in Table 6.3.

↑ Output

↓ Output

Output

Table 6.3 Project Success Rates



However if censoring is switched off using `\StopCensoring`, the redacted text will now be replaced by an underlined blank space (in the case of `\censor*`) or an unfilled rectangle (in the case of `\censorbox*`):

6.4 Confidentiality

Following the research group's unsuccessful attempt to create _____, they will now be working on a new _____. The success rate of previous projects is shown in Table 6.4.

↑ Output

↓ Output

Table 6.4 Project Success Rates

Output



In your secure environment, you can change the definitions of the macros for the redacted material:

↑ Input

```
\newcommand*\OldProject{%
\censor{mind}-\censor{controlling} \censor{cookies}}
```

6.4 Confidentiality

```
\newcommand*{\NewProject}{\censor{ray} \censor{gun}}
\newcommand*{\SuccessRates}{%
\censorbox{%
\begin{tabular}{lc}
\bfseries Project & \bfseries Success Rate \\
Mind-controlling Cookies & 2\%\\
Telepathic Cakes & 1\%\\
Exploding Chocolates & 25\%
\end{tabular}}%
}%
}
```

↓ Input

There may be minor discrepancies in the formatting caused by approximate measurements.

It's possible you may not want to keep editing the definitions of commands like the example `\OldProject`, `\NewProject` and `\SuccessRates`, if you keep transferring your document between a secure and an insecure location. Instead, it's better to keep the real definitions of these commands (using the unstarred versions of `\censor` and `\censorbox`) in a separate file, and only input the file if it exists. Then, when you transfer your document to an insecure location, make sure you don't also transfer this sensitive file.

6.4 Confidentiality

EXAMPLE 33. REDACTION

Suppose you have a file called `definitions.tex` that contains:

```
\newcommand*\OldProject{%
  \censor{mind}-\censor{controlling} \censor{cookies}}
\newcommand*\NewProject{\censor{ray} \censor{gun}}
\newcommand*\SuccessRates{%
  \censorbox{%
    \begin{tabular}{lc}
      \bfseries Project & \bfseries Success Rate \\
      Mind-controlling Cookies & 2\%\\
      Telepathic Cakes & 1\%\\
      Exploding Chocolates & 25\%
    \end{tabular}}}
```

↑ Input

↓ Input

This is your secret file that shouldn't leave your secure location. In your main document you can test for a file's existence, and only input it if it exists, using:

6.4 Confidentiality

\InputIfFileExists{⟨file⟩}{⟨true part⟩}{⟨false part⟩}

Definition

This tests if the file named ⟨file⟩ exists (the .tex extension may be omitted). If the file exists, this command does ⟨true part⟩ and then loads ⟨file⟩. If the file doesn't exist, this command just does ⟨false part⟩. So the main document can include this command to determine whether to use the starred or unstarred versions:

↑ Input

```
\documentclass[captions=tableheading]{scrartcl}

\usepackage{censor}

\InputIfFileExists{definitions}{}%
{%
    \newcommand*\OldProject{\censor*{4}-\censor*{11} \censor*{7}}
    \newcommand*\NewProject{\censor*{3} \censor*{3}}
    \newcommand*\SuccessRates{\censorbox*{40}{4}{0}}
}

\begin{document}
```

6.4 Confidentiality

Following the research group's unsuccessful attempt to create \OldProject, they will now be working on a new \NewProject. The success rate of previous projects is shown in Table~\ref{tab:success}.

```
\begin{table}
  \caption{Project Success Rates}
  \label{tab:success}
  \centering
  \SuccessRates
\end{table}

\end{document}
```

↓ Input

You can [download](#) or [view](#) this example.

6.4.2 Watermarks

There are a number of packages that enable you to place some text (such as "CONFIDENTIAL" or "DRAFT") across the background of every page.

[FAQ:
'Watermarks' on
every page]

6.4 Confidentiality

In addition to these packages, the pdftk application described in §2.3, can also be used for this purpose. CTAN has several topics that cover this area: [watermark](#), [background](#) and [decoration](#). This section will only discuss two packages: `xwatermark` (which extends the `draftmark` and `watermark` packages) and `background`. The `background` package is simpler to use. The `xwatermark` package is more flexible.

The background Package

The `background` package [56] provides a way of specifying a watermark using the `tikz` package (part of the `pgf` bundle [102]). By default, the package will use the word “Draft” as a watermark, but this can be changed. Options can either be set via the package option list or via:

`\backgroundsetup{\langle options \rangle}`

Definition

where `\langle options \rangle` is a [key=value list](#). The `background` package requires two L^AT_EX calls during the document build. Available options are:

- pages This key specifies whether the watermark should appear on all or some of the pages. Available values are: `all` or `some`. If the value `some` is specified, you must use the command `\BgThisPage` on the pages where the watermark is required. If

6.4 Confidentiality

the value `all` is specified, you can use the command `\NoBgThisPage` on the pages where you don't want a watermark. (Don't use this command in two-column mode.)

- firstpage** This is a [boolean key](#). If `true`, the watermark only appears on the first page. (The default is `false`.)
- placement** This key specifies the placement of the watermark and may take one of the values: `center` (default), `top` or `bottom`.
- contents** This key specifies the material used for the watermark. The value may be just text (such as "Confidential") or an image included using [`\includegraphics`](#). If the value of this key contains commands, the key may only be used within [`\backgroundsetup`](#) not as a package option.
- color** The watermark colour. (The `background` package automatically loads the `xcolor` package [41].)
- angle** This key specifies the angle of rotation.
- opacity** This key indicates the transparency. The value may be a number from 0 (full transparency) to 1 (no transparency). Note that

6.4 Confidentiality

PostScript doesn't support transparency, so if you use `latex` and `dvips`, the watermark won't show any transparency in the PostScript file.

- scale** This key specifies the scale factor.
- position** This key can be used to adjust the position of the watermark. The value should be in TikZ coordinate form (see the pgf manual [102]).
- anchor** This key specifies the TikZ anchor for the watermark. (See the pgf manual [102].)
- hshift** This key specifies the horizontal shift.
- vshift** This key specifies the vertical shift.

EXAMPLE 34. WATERMARKS (background PACKAGE)

This example uses the `lipsum` package [33] to generate dummy text to pad out the document.

6.4 Confidentiality

↑ Input

```
\documentclass[a4paper]{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\usepackage{background}
\usepackage{lipsum}

\backgroundsetup{contents={CONFIDENTIAL}}


\begin{document}

\lipsum[1-55]

\end{document}
```

↓ Input

The first page of this document is shown in [Figure 6.4](#). You can [download](#) or [view](#) this example.

6.4 Confidentiality

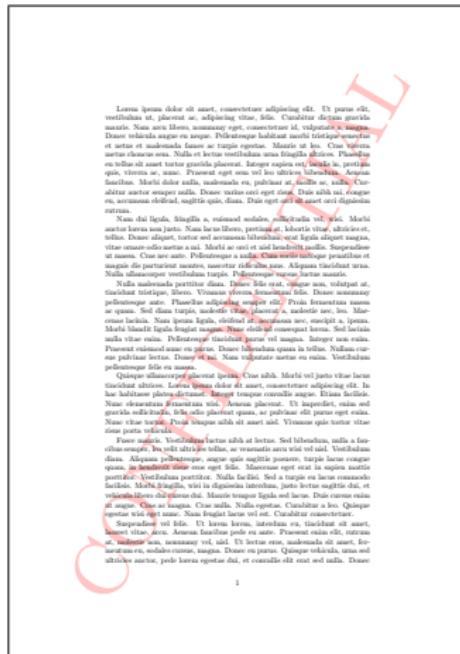


Figure 6.4 Sample Watermark (background package)

The xwatermark Package

The `xwatermark` package [60] provides a way of adding either text or graphics as watermarks in the background or foreground of selected pages. In most cases, more than one `LATEX` run will be required on your document. You will also mostly need to load a colour package, such as `xcolor` [41]. The `xwatermark` package can also be used to tile marks across the page, but that option isn't covered here. See the `xwatermark` documentation for details.

There are a number of package options for `xwatermark`. Only a subset of them are listed here:

`printwatermark` This is a [boolean key](#) that determines whether or not the watermarks should be printed. The default value is `true`.

`picontoptext` This is a [boolean key](#) that determines whether the picture watermark should be placed on top of the text watermark where they occur on the same page. The default value is `true`.

`showpagecenter` This is a [boolean key](#) that determines whether the centre of the paper should be shown with a marker. The default value is `false`.

6.4 Confidentiality

`disablegeometry` This is a [boolean key](#). If `true`, this indicates that the page layout settings by the `geometry` package [110] should be disabled. The default value is `false`. (This option is less likely to be needed with newer versions of `geometry`.)

See the `xwatermark` package documentation for details of the other package options.

A new watermark is defined using:

`\newwatermark[<options>]{<mark>}`

[Definition](#)

where `<mark>` is the watermark text. This may be empty if you want an image for the watermark. There is a starred variant of this command that puts the watermark in the foreground instead of the background. There is also a prime variant, but that's not covered here. See the `xwatermark` documentation [60] for further details.

The optional argument, `<options>`, is a [key=value list](#) of options. For brevity, this section only covers a small subset of those options. See the `xwatermark` documentation for further details. The option list must contain the page or page set on which the watermark is to be displayed. The page specifier keys are as follows:

`allpages` This key doesn't have a value. It indicates that the watermark should be displayed on all pages.

6.4 Confidentiality

- oddpages** This key doesn't have a value. It indicates that the watermark should be displayed on all odd pages.
- evenpages** This key doesn't have a value. It indicates that the watermark should be displayed on all even pages.
- firstpage** This key doesn't have a value. It indicates that the watermark should only be displayed on the first page.
- lastpage** This key doesn't have a value. It indicates that the watermark should only be displayed on the last page.
- page** The value of this key should be a page number to indicate that the page on which the watermark should be displayed. For example, `page=4` indicates that the watermark should be displayed on page 4.
- pages** The value of this key should be a range in the form $\langle n \rangle - \langle m \rangle$ to indicate that the watermark should be displayed on pages $\langle n \rangle$ to $\langle m \rangle$, inclusive. For example, `pages=4-10` indicates that the watermark should be displayed on pages 4 to 10.
- pagex** The value of this key should be a [comma-separated list](#) of page numbers on which the watermark should be displayed. For

6.4 Confidentiality

example, `pagex={1,3,7}` indicates that the watermark should be displayed on pages 1, 3 and 7.

The text for the watermark may also be specified in `<options>` using the `textmark={<mark>}` key. The watermark can be scaled, rotated or translated using the following keys:

`scale` The value of this key is the scaling factor.

`angle` The value of this key is the angle of rotation.

`xpos` The value of this key specifies the horizontal position of the watermark relative to the centre of the page.

`ypos` The value of this key specifies the vertical position of the watermark relative to the centre of the page.

The formatting of the watermark can be adjusted using the following keys:

`textalign` The value of this key indicates the horizontal alignment of the watermark and may be one of: `center` (default), `left`, `right` or `justified`.

`fontfamily` The value of this key is the name of the font family (as supplied to `\fontfamily`). For example, `fontfamily=pbk` indicates Bookman.

6.4 Confidentiality

fontseries The value of this key is the name of the font series (as supplied to `\fontseries`). For example, `fontseries=b` indicates bold.

fontsize The value of this key is the font size. The default is 1 cm.

textcolor The value if this key is the colour of the watermark text.

If an image is required, the following keys specify the image file information:

picfile The value of this key is the image filename. This should be in the same form as the mandatory argument of `\includegraphics`, so remember to use forward slashes for the directory divider even if you are building your document on a Windows operating system.

picfileext The value of this key is the image file extension (such as pdf or png). Note that you have to use `picfileext` if you use `picfile`.

picscale The value of this key is the scaling factor to apply to the image.

picangle The value of this key is the angle of rotation to apply to the images.

6.4 Confidentiality

There are other keys not covered here. See the `xwatermark` documentation [60] for further details.

EXAMPLE 35. WATERMARKS (`xwatermark` PACKAGE)

As with Example 34, this example uses the `lipsum` package [33] to generate dummy text to pad out the document.

↑ Input

```
\documentclass[a4paper]{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\usepackage{xcolor}
\usepackage{xwatermark}
\usepackage{lipsum}

\newwatermark[%  

    allpages,% show on all pages  

    fontfamily=pbk,% use Bookman font family  

    angle=55,% rotate by 55 degrees  

    scale=2.75,% scale by 2.75
```

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
xpos=-1cm,% shift by 1cm to the left  
ypos=1cm% shift up by 1cm  
]{SAMPLE WATERMARK}
```

```
\begin{document}  
\lipsum[1-55]  
  
\end{document}
```

↓ Input

The first page of this document is shown in [Figure 6.5](#). You can [download](#) or [view](#) this example.

6.5 Typesetting Legal Documents (Numbered Paragraphs)

There are a number of packages listed on the [legal topic](#) page for typesetting legal documents, however they are mostly for particular areas, such as German legal texts or US patent applications. Since they are too specific for a general guide, this section just looks at how to produce the paragraph

6.5 Typesetting Legal Documents (Numbered Paragraphs)



Figure 6.5 Sample Watermark (xwatermark package)

6.5 Typesetting Legal Documents (Numbered Paragraphs)

numbering styles used in many legal contexts, such as terms and conditions of sale. There are two main types of paragraph numbering styles: hierarchical and non-hierarchical. The first is covered in §6.5.1 and the second is covered in §6.5.2.

6.5.1 ■ Hierarchical Paragraph Numbering

Suppose you want a document that looks something like the following:

1. Information About Us

↑ Output

- 1.1. This website is run by the Secret Lab of Experimental Stuff (“We”, “Our”). We operate from the University of Somewhere.

2. Our Products

- 2.1. All Products shown on our site are subject to availability.
- 2.2. You may only purchase our products if you are at least 18 years old.

3. Refunds

6.5 Typesetting Legal Documents (Numbered Paragraphs)

3.1. You are entitled to a refund unless:

- 3.1.1. you have eaten the mind-controlling cookies;
- 3.1.2. you have thrown the exploding chocolates;
- 3.1.3. you have used the ray gun as:
 - 3.1.3.1. a table chock;
 - 3.1.3.2. a weapon unless:
 - 3.1.3.2.1. you have a ray gun permit;
 - 3.1.3.2.2. you are an extraterrestrial.

↓ Output

The simplest way of achieving this is to use the `enumerate` environment (described in [Volume 1 \[93, §4.4.2\]](#)) and redefine the way the counters are displayed. By default, L^AT_EX allows up to four nested `enumerate` environments. Each level has a separate counter: `enumi`, `enumii`, `enumiii` and `enumiv`. Recall from [Volume 1 \[93, §11\]](#) that the displayed value of a counter is governed by the command:

6.5 Typesetting Legal Documents (Numbered Paragraphs)

\the<counter>

Definition

(For example, \theenumi.) The default formats for each level use: \arabic, \alph, \roman and \Alph. The format used if the items are cross-referenced using the \label/\ref mechanism prefixes \the<counter> with:

\p@<counter>

Definition

For example, \p@enumiv\theenumiv. The prefix is ignored if \p@<counter> doesn't exist (so just \the<counter> is used).

The above \the<counter> and \p@<counter> are general counter-related commands. In addition, for the enumerate environment, there are also commands that determine the item label formats:

\label<counter>

Definition

(For example, \labelenumi for the first level.)

EXAMPLE:

Suppose you want to have a lower case letter for the third level enumerate items:

\renewcommand*\theenumiii{\alph{enumiii}}

Input

and you want the label for the third level enumerate items to use parentheses:

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\renewcommand*{\labelenumiii}{(\theenumiii)}
```

Input

but when you cross-reference a third level enumerate item, `\ref` should use the format `(level1).(level2).(level3)`:

```
\renewcommand*{\p@enumiii}{\theenumi.\theenumii.\theenumiii}
```

Input

or, similarly setting `\p@enumii`:

```
\renewcommand*{\p@enumii}{\theenumi.}  
\renewcommand*{\p@enumiii}{\p@enumii\theenumiii}
```

↑ Input

↓ Input

EXAMPLE 36. NESTED ENUMERATION

To reproduce the format shown at the start of this section, the standard `\section` command can be used for the first level (“Information About Us”, “Our Products” and “Refunds”). The levels below this can be created using nested `enumerate` environments, where the counter formats are defined as:

↑ Input

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\renewcommand*{\theenumi}{\thesection.\arabic{enumi}}
\renewcommand*{\theenumii}{\theenumi.\arabic{enumii}}
\renewcommand*{\theenumiii}{\theenumii.\arabic{enumiii}}
\renewcommand*{\theenumiv}{\theenumiii.\arabic{enumiv}}
```

↓ Input

and the labels are defined as:

```
\renewcommand*{\labelenumi}{\theenumi.}
\renewcommand*{\labelenumii}{\theenumii.}
\renewcommand*{\labelenumiii}{\theenumiii.}
\renewcommand*{\labelenumiv}{\theenumiv.}
```

↑ Input

Since all the counter formats (`\theenumi`, ..., `\theenumiv`) are now hierarchical, the **internal commands** used as a prefix by the cross-referencing mechanism need to be defined to do nothing:

```
\renewcommand*{\p@enumi}{}
\renewcommand*{\p@enumii}{}
```

↑ Input

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\renewcommand*{\p@enumiii}{}  
\renewcommand*{\p@enumiv}{}  
↓ Input
```

The text can now be formatted as follows:

```
\section{Information About Us}  
  
\begin{enumerate}  
  \item This website is run by the Secret Lab of Experimental Stuff  
    ('`We'', ``Our''). We operate from the University of Somewhere.  
\end{enumerate}  
  
\section{Our Products}  
  
\begin{enumerate}  
  \item All Products shown on our site are subject to availability.  
  
  \item You may only purchase our products if you are at least  
    18 years old.  
\end{enumerate}
```

```
\section{Refunds}

\begin{enumerate}
\item You are entitled to a refund unless:
\begin{enumerate}
\item you have eaten the mind-controlling cookies;
\item you have thrown the exploding chocolates;
\item you have used the ray gun as:
\begin{enumerate}
\item a table chock;
\item a weapon unless:
\begin{enumerate}
\item \label{permit}you have a ray gun permit;
\item you are an extraterrestrial.
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}
```

↓ Input

I've labelled one of the items using `\label`, so I can reference it using:

6.5 Typesetting Legal Documents (Numbered Paragraphs)

See clause~\ref{permit}.

Input

which produces:

See clause 3.1.3.2.1.

Output

You can [download](#) or [view](#) a complete document.

☰ Delving Deeper

If your document is complicated enough to require deeper levels than the default maximum of four, it's possible to extend the maximum `enumerate` depth. In order to do this, it's necessary to:

1. Modify the `enumerate` environment to allow more levels. This may additionally require modifying the underlying generic `list` environment that only allows a maximum of six nested list environments.
2. Define a new `enum<n>` counter (using `\newcounter`) for each level $\langle n \rangle$, where $\langle n \rangle$ is the lower case Roman numeral representing the level index. (For example, `enumv` for the fifth level).

6.5 Typesetting Legal Documents (Numbered Paragraphs)

3. Define a new `\labelenum{n}` command for the item label for each level $\langle n \rangle$.
4. Define a new `\leftmargin{n}` length (using `\newlength` and `\setlength`) for the left margin for each level $\langle n \rangle$.
5. Define a new `@list{n}` command that sets up the margin for each level $\langle n \rangle$.

The counters and label commands are the easy part. For example, to allow a maximum of six levels, counters and label commands need to be defined for levels 5 (v) and 6 (vi):

```
\newcounter{enumv}[enumiv]
\newcounter{enumvi}[enumv]
\newcommand*{\labelenumv}{\theenumv.}
\newcommand*{\labelenumvi}{\theenumvi.}
```

↑ Input

↓ Input

(If necessary, you can also redefine the counter prefixes `\p@{<counter>}`.)

The left margin lengths are already provided up to six levels, but supposing you needed a seventh (vii), this would be done using:

6.5 Typesetting Legal Documents (Numbered Paragraphs)

↑ Input

```
\newlength{\leftmarginvii}  
\setlength{\leftmarginvii}{15pt}
```

↓ Input

(Change the length as required.)

The `\@list{n}` commands are more complicated. These commands need to set the length `\leftmargin` to the current level's left margin `\leftmargin{n}` and set the `\labelwidth` length to `\leftmargin{n}` less the value of `\labelsep`. Again, there are already up to six levels provided, but supposing you needed a seventh (vii), this would be done using:

↑ Input

```
\newcommand*{\@listvii}{%  
  \setlength{\leftmargin}{\leftmarginvii} %  
  \setlength{\labelwidth}{\leftmarginvii} %  
  \addtolength{\labelwidth}{-\labelsep} %  
}
```

↓ Input

6.5 Typesetting Legal Documents (Numbered Paragraphs)

Modifying the `enumerate` environment is slightly harder. This can be done using `\renewenvironment` (described in [Volume 1 \[93, §10.1\]](#)) however, in this case, only the beginning of the environment needs changing. The `\begin{<env-name>}` command works by (amongst other things) using the command `\<env-name>` so `\begin{enumerate}` calls the command `\enumerate`, and it's this command that needs modifying. Recall from [§2.1.1](#) that you can find out the definition of a command using either `\show` in your document or using the `texdef` script. If I run:

```
texdef -t latex enumerate
```

Shell

I get the response (tidied up a bit for legibility):

```
\enumerate:  
macro:->\ifnum \@enumdepth >\thr@@  
    \@toodeep  
\else  
    \advance\@enumdepth\@ne  
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}-%  
    \expandafter\list\csname label\@enumctr\endcsname{%-  
        \usecounter\@enumctr  
        \def\makelabel##1{\hss\llap{##1}}%  
    }%
```

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\fi
```

The key part here is the \TeX conditional:

```
\ifnum \@enumdepth > \thr@@
```

This tests if the number stored in the `\@enumdepth` register is greater than `\thr@@` (which is defined in the \LaTeX kernel to have the value 3). So the new definition of `\enumerate` needs to change `\thr@@` to one less than the new maximum.

Since this code contains [internal commands](#), the new definition should either be placed in a class or package or, if used in the document, be placed between `\makeatletter` and `\makeatother`. Therefore to set the maximum to six levels:

```
\makeatletter  
  
\renewcommand*{\enumerate}{%  
  \ifnum \@enumdepth > 5  
    \atodeep  
  \else  
    \advance\@enumdepth\@ne  
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
```

↑ Input

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\expandafter\list\csname label\@enumctr\endcsname{%
  \usecounter\@enumctr
  \def\makelabel##1{\hss\llap{##1}}%
}%
\fi
}

\makeatother
```

↓ Input

If you really do need more than six levels (although I hope you don't), you similarly need to modify `\list`, which by default tests if `\@listdepth` is greater than five.

EXERCISE 20. EXTENDING THE MAXIMUM enumerate DEPTH

For this exercise, extend the maximum `enumerate` depth to 6 levels (as described above), so that you can create the following:

1. Information About Us

↑ Output

- 1.1. This website is run by the Secret Lab of Experimental Stuff ("We", "Our"). We operate from the University of Somewhere.

6.5 Typesetting Legal Documents (Numbered Paragraphs)

2. Our Products

- 2.1. All Products shown on our site are subject to availability.
- 2.2. You may only purchase our products if you are at least 18 years old.

3. Refunds

- 3.1. You are entitled to a refund unless:
 - 3.1.1. you have eaten the mind-controlling cookies;
 - 3.1.2. you have thrown the exploding chocolates;
 - 3.1.3. you have used the ray gun as:
 - 3.1.3.1. a table chock;
 - 3.1.3.2. a weapon unless:
 - 3.1.3.2.1. you have a ray gun permit;
 - 3.1.3.2.2. you are an extraterrestrial and:
 - 3.1.3.2.2.1. are not a resident of the planet Earth;
 - 3.1.3.2.2.2. have a licence under Galactic Treaty 1024, unless:
 - 3.1.3.2.2.2.1. you come under Article 24, or

3.1.3.2.2.2. you live on Saturn.

↓ Output

The choice of document class is up to you. For example, you can use the article or scrartcl classes. (If you use scrartcl, the class option numbers=endperiod will display a full stop after the section numbers.) You can [download](#) or [view](#) a solution.

6.5.2 ■ Non-Hierarchical Paragraph Numbering

Suppose now that instead of the hierarchical structure illustrated [above](#), you simply want all your paragraphs numbered sequentially. TeX has a mechanism for specifying code that should be performed at the start of each paragraph:

`\everypar{\langle code\rangle}`

Definition

The numbering can be dealt with using a counter. For example, I could define a new counter called, say, para:

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\newcounter{para}
```

↑ Input

and I could define a command called, say, `\numberedparagraph`:

```
\newcommand*\numberedparagraph{%
  \refstepcounter{para}\thepara.\space
}
```

↑ Input

↓ Input

This command needs to go at the start of each paragraph, but it's rather tiresome to do this manually, so `\everypar` can be used instead. For example (using the `lipsum` package [33] to generate dummy text):

```
\everypar{\numberedparagraph}
\lipsum[1-3]
```

↑ Input

↓ Input

produces:

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

6.5 Typesetting Legal Documents (Numbered Paragraphs)

3. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

↓ Output

Since the para counter is incremented using `\refstepcounter`, the paragraphs can be cross-referenced using the standard `\label/\ref` mechanism. If the paragraph numbering needs to be reset every page, you can specify the page counter as the “master” counter when you define the para counter:

[FAQ: Master and slave counters]

`\newcounter{para}[page]`

Input

⚠ In general you need to be careful about using page as a master counter, but in this case it's not a problem as this new para counter only gets incremented at the beginning of a paragraph so it doesn't conflict with TeX's output routine.

6.5 Typesetting Legal Documents (Numbered Paragraphs)

There is, however, a problem: some commands, such as the section commands, use `\everypar` to reset the paragraph behaviour. So, for example, a `\chapter` or `\section` command will override an earlier use of `\everypar`. It's also unlikely that you'll want the chapter and section headings to have a paragraph number. This last issue is easily dealt with by hooking into the sectioning commands using one of the `etoolbox` commands described in §2.1.2:

```
\preto\chapter{\everypar{}}
\preto\section{\everypar{}}
\preto\subsection{\everypar{}}
\preto\subsubsection{\everypar{}}
\preto\paragraph{\everypar{}}
\preto\ subparagraph{\everypar{}}
```

↑ Input

↓ Input

The first issue, redoing `\everypar{\numberedsection}` after every sectioning command, is more complicated. Most classes use:

`\@afterheading`

Definition

within the definition of the sectioning commands. This command uses `\everypar` to suppress the indentation of the first paragraph following the

6.5 Typesetting Legal Documents (Numbered Paragraphs)

heading, and within the argument of `\everypar` there is another `\everypar` that resets the paragraph hook back to empty, which ensures that subsequent paragraphs are indented.

As in [the previous section](#), either the `\show` command or the `texdef` script can be used to show the original definition of `\@afterheading`. For example, if I run:

```
texdef -t latex @afterheading
```

Shell

I get (reformatted for clarity):

```
\@afterheading:  
macro:->\nobreaktrue  
\everypar{  
  \if@nobreak  
    \nobreakfalse  
    \clubpenalty@\M  
    \if@afterindent  
    \else  
      {\setbox \z@ \lastbox }%  
    \fi  
  \else  
    \clubpenalty\clubpenalty
```

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\everypar{}%
\fi
}
```

So `\@afterheading` can be redefined to use our new `\numberedparagraph` command:

↑ Input

```
\renewcommand{\@afterheading}{%
 \@nobreaktrue
 \everypar{%
 \if@nobreak
   \@nobreakfalse
   \clubpenalty\@M
 \if@afterindent
 \else
   {\setbox\z@\lastbox}%
 \fi
 \else
   \clubpenalty\@clubpenalty
 \everypar{\numberedparagraph}%
 <- modification
 \fi
}
```

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\numberedparagraph% <- modification  
}%  
}
```

↓ Input

Remember that this code uses **internal commands**, so either use `\makeatletter` and `\makeatother` or place the code in a package or class.

EXERCISE 21. NUMBERED PARAGRAPHS

Modify the following document code so that the paragraphs are automatically numbered:

```
\documentclass{article}  
  
\usepackage{lipsum}% dummy text  
  
\author{Some One}  
\title{Numbered Paragraphs Example}  
  
\begin{document}  
\maketitle
```

↑ Input

6.5 Typesetting Legal Documents (Numbered Paragraphs)

```
\section{Sample Section}

\lipsum[1-4]

Some\label{sample} sample text.

\lipsum[5-10]

\subsection{Sample Subsection}

\lipsum[11-15]

\section{Another Section}

\lipsum[16-30]

\end{document}
```

↓ Input

Also, add a cross-reference to the paragraph labelled `sample`.

You can [download](#) or [view](#) a solution.

FOR THE MORE ADVENTUROUS:

Modify the definition of `\numberedparagraph` so that it uses

6.5 Typesetting Legal Documents (Numbered Paragraphs)

\marginpar[*left*]{*text*}

Definition

to put the paragraph numbers in the margins.

7. DATES AND TIMES

There's a large list of packages related to dates and times in the “[Calendars, Date and Time](#)” of the \TeX Catalogue Topic Index [25] and on the [calendar topic](#), [timetable topic](#) and [date-time topic](#) pages.

However, once you discount the plain \TeX macros, the old $\text{\LaTeX}2.09$ styles, the packages that aren't included in \TeX Live or MiK \TeX , packages that don't have English documentation and packages that don't have proper documentation (just an example file or plain text file), then the list becomes much smaller and can be divided into those that provide commands to: compute dates; display calendars or timetables; display specific dates or times; or display the current time or the current date in a particular format (by redefining `\today`). Some of the packages cover more than one of these categories.

The `datetime` package, which can be used to display formatted dates and the current time, was described in [Volume 1](#) [93, §4.2.1] and so is not covered here. Similarly the `babel` package [7], which also redefines `\today`, was described in [Volume 1](#) [93, §5.8]. Since I started writing this book, I have replaced `datetime` with `datetime2`, which uses some of the code described in

this chapter, so `datetime2` is briefly introduced in §7.1.

⚠ In this book, “current time” and “current date” (or `\today`) indicate the time or date of the TeX run that created the resulting PDF (or DVI) document. If you actually want a real time clock in your document, you can use the `tdclock` package [70] however this will only work if your PDF viewer not only supports JavaScript (such as Adobe Reader) but also has JavaScript enabled.

Surprisingly none of the above CTAN date-related topics include the `pgfcalendar` package that comes with the `pgf` bundle [102]. This is a useful date utility package that can be used for computing dates by adding or subtracting days from a given date. It also has commands that display month and week day names, so it can be used for parsing and formatting dates. In addition, since the `pgf` package’s main function is graphical, `pgfcalendar` can be used to display calendars.

This chapter is arranged as follows:

- §7.1 briefly introduces the `datetime2` package.
- §7.2 describes the date utility commands of the `pgfcalendar` package that can: parse a date, or a date offset from another date, and convert it to a Julian day number; convert a Julian day number into an ISO-date; determine the week day from a Julian day number;

7.1 The `datetime2` Package

test a date (for example, test if the date is on a particular week day, or is earlier or later than another date).

- §7.3 describes how to display a date using your own preferred format.
- §7.5 describes how to use the `pgfcalendar` package to display a calendar.
- §7.4 describes how to parse and display times.

7.1 The `datetime2` Package

The `datetime2` package has replaced the now obsolete `datetime` package. This section is just a brief introduction, see the user guide [101] for other commands and settings not described here.

The simplest use of this package is:

```
\documentclass{article}  
\usepackage{datetime2}
```

↑ Input

7.1 The `datetime2` Package

```
\begin{document}  
This PDF was created on \today.  
\end{document}
```

↓ Input

This produces

This PDF was created on 2015-09-08.

Output

This is the default date format. If you want the full date, time and time zone, you can use

`\DTMnow`

Definition

instead of `\today`. This will display the date and time in the form

2015-09-08 11:17:58+01:00

Output

Note that X_EL_AT_EX doesn't provide the time zone information so you will need to use PDFL_AT_EX or L_AU_LE_AT_EX if you want this.

If you want to use a regional format, you can specify the region in the package option. For example:

↑ Input

7.1 The `datetime2` Package

```
\documentclass{article}
\usepackage[en-GB]{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

↓ Input

This produces:

This PDF was created on 8th September 2015 11:17am BST.

Output

If you want to pick up the regional setting from babel you can use the `useregional` option:

```
\documentclass[british]{article}
\usepackage[babel]
\usepackage[useregional]{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

↑ Input

7.1 The *datetime2* Package

If you prefer a numeric regional format you can use `useregional=numeric` instead.

 You must additionally install the appropriate language module, for example, `datetime2-english` [100] to enable the regional support. If you want to display the day of the week name, you can use the `showdow` package option, but this isn't available for all regions. You will need to check the documentation for the relevant module to find out if it's supported.

If you want to display a particular date (without the time) in the current style you can use

`\DTMdate{<date>}`

Definition

where `<date>` is given in the format `<yyyy>-<mm>-<dd>`. For example:

`\DTMdate{2015-09-08}`

Input

You can save a date and time for later use with:

`\DTMsavetimestamp{<name>}{<data>}`

Definition

where `<name>` is a unique label that identifies this date and time, and `<data>` is in the format

`<YYYY>-<MM>-<DD>T<hh>:<mm>:<ss><zone>`

7.1 The *datetime2* Package

where $\langle\text{YYYY}\rangle$ is the year, $\langle\text{MM}\rangle$ is the month number, $\langle\text{DD}\rangle$ is the day of the month, $\langle\text{hh}\rangle$ is the hour (24), $\langle\text{mm}\rangle$ is the minute value, $\langle\text{ss}\rangle$ is the second value and $\langle\text{zone}\rangle$ is the time zone, which may be either Z or in the format $\langle\text{TZh}\rangle:\langle\text{TZm}\rangle$ where $\langle\text{TZh}\rangle$ is the hour offset and $\langle\text{TZm}\rangle$ is the minute offset. The argument $\langle\text{data}\rangle$ may also be a control sequence that expands (one level) to the required format.

Alternatively, you can just save the date with:

`\DTMsavedate{\langle name \rangle}{\langle date \rangle}`

Definition

where the $\langle\text{date}\rangle$ is in the format $\langle\text{YYYY}\rangle-\langle\text{DD}\rangle-\langle\text{MM}\rangle$ and $\langle\text{name}\rangle$ is again a label.

A previously saved date and time can be displayed in the current style using:

`\DTMuse{\langle name \rangle}`

Definition

Just the date can be displayed using:

`\DTMusedate{\langle name \rangle}`

Definition

and just the time with:

`\DTMusetime{\langle name \rangle}`

Definition

In all cases, $\langle\text{name}\rangle$ is the label identifying the date or time stamp.

7.1 The `datetime2` Package

EXAMPLE 37. DISPLAYING DATES AND TIMES (datetime2 PACKAGE)

Here's an example that uses the en-GB style:

```
\documentclass{article}

\usepackage[en-GB]{datetime2}

\newcommand*\{\DateStamp}{2015-11-28T20:13:04Z}
\DTMsavetimestamp{mydate}{\DateStamp}
\DTMsavetimestamp{mydate2}{2014-06-01T09:01:58+01:00}
```

↑ Input

```
\begin{document}
```

```
Now: \DTMnow.
```

```
Saved: \DTMuse{mydate}; \DTMuse{mydate2}.
```

```
\end{document}
```

↓ Input

This produces

7.2 The *pgfcalendar* Package Utility Commands

Now: 8th September 2015 11:25am BST.

↑ Output

Saved: 28th November 2015 8:13pm GMT; 1st June 2014 9:01am BST.

↓ Output

You can [download](#) or [view](#) this document.

 Be careful if you are using babel with the shorthands on as this may change the `category code` of characters such as : and - which will cause the date or time parsing in commands like `\DTMsavetimestamp` to fail. You may need to temporarily switch off the shorthands. See the babel manual [7] for further details.

7.2 ■ The *pgfcalendar* Package Utility Commands

The *pgfcalendar* package may be used independently of the *pgf* package, but if used without *pgf* (or *tikz*) the *pgfkeys* package also needs to be loaded:

7.2 The *pgfcalendar* Package Utility Commands

```
\usepackage{pgfkeys}  
\usepackage{pgfcalendar}
```

↑ Input

or

```
\usepackage{pgfkeys,pgfcalendar}
```

Input

In the command definitions below, *<date>* indicates a date specified using one of the following formats:

- A specific date:

<year>-<month>-<day>

- The last day of a particular month:

<year>-<month>-last

- An increment from a given date:

<year>-<month>-<day>+<increment>

or

<year>-<month>-last+<increment>

7.2 The *pgfcalendar* Package Utility Commands

Where $\langle year \rangle$ is the year (for example, 2014 or `\year` for the current year); $\langle month \rangle$ is the month number (for example, 6 for June or `\month` for the current month) and $\langle day \rangle$ is the day number (for example, 21 for the twenty-first of the month or `\day` for the current day). The $\langle increment \rangle$ (in days) may be either a positive or negative number. If negative the leading + is still required. For example, 2014-6-last+-4 means four days before the last day of June.

The *pgfcalendar* provides the commands:

`\pgfcalendardatejulian{\langle date \rangle}{\langle register \rangle}`

Definition

This converts a Gregorian date into the Julian day number and stores the result in $\langle register \rangle$, which must be a TeX register (*not* a L^AT_EX counter).

EXAMPLE:

(Recall `\newcount` from §2.1.3.)

```
\newcount\mycount  
\pgfcalendardatejulian{2014-03-18+2}{\mycount}  
\the\mycount
```

↑ Input

↓ Input

7.2 The *pgfcalendar* Package Utility Commands

produces:

2456737

Output

`\pgfcalendarjuliantodate{\langle Julian day \rangle}{\langle year cs \rangle}{\langle month cs \rangle}{\langle day cs \rangle}`

Definition

This converts a Julian day number to an ISO-date and stores the resulting year, month and day-of-month numbers in the control sequences `\langle year cs \rangle`, `\langle month cs \rangle` and `\langle day cs \rangle`.

EXAMPLE:

```
\pgfcalendarjuliantodate{2456737}{\theyear}{\themonth}{\theday}
```

↑ Input

produces:

2014/03/20

Output

7.2 The *pgfcalendar* Package Utility Commands

`\pgfcalendarjuliantoweekday{\langle Julian day\rangle}{\langle register\rangle}`

Definition

This converts a Julian day number to a week day number, where 0 indicates Monday, 1 indicates Tuesday, etc. The result is stored in the TeX register specified by *<register>*.

EXAMPLE:

```
\newcount\mycount  
\pgfcalendarjuliantoweekday{2456737}{\mycount}  
\the\mycount
```

↑ Input

↓ Input

produces:

3

Output

(which indicates Thursday).

`\pgfcalendarifdate{\langle date\rangle}{\langle test\rangle}{\langle true code\rangle}{\langle false code\rangle}`

Definition

This tests the given date and does *<true code>* if the test succeeds otherwise it does *<false code>*. The *<test>* may be one of the following key words:

7.2 The *pgfcalendar* Package Utility Commands

- all Always yields true.
- Monday True if the date is a Monday.
- Tuesday True if the date is a Tuesday.
- Wednesday True if the date is a Wednesday.
- Thursday True if the date is a Thursday.
- Friday True if the date is a Friday.
- Saturday True if the date is a Saturday.
- Sunday True if the date is a Sunday.
- workday True if the date occurs from Monday to Friday, inclusive.
- weekend True if the date is a Saturday or Sunday.

Or the $\langle test \rangle$ may be a comparison against a reference, which may be an ISO date in the form $\langle yyyy \rangle\text{-}\langle mm \rangle\text{-}\langle dd \rangle$ (for example, 2014-03-20) or with the year missing $\langle mm \rangle\text{-}\langle dd \rangle$ (for example, 03-20):

7.2 The *pgfcalendar* Package Utility Commands

- `equals=<reference>`

True if `<date>` is the same date as reference (where the year is specified in `<reference>`) or has the same month and day parts as the reference (where the year is omitted from `<reference>`).

- `at least=<reference>`

True if `<date>` is equal to `<reference>` or is a later date than `<reference>`. If the year is omitted from `<reference>`, only the month and day in `<date>` form part of the test.

- `at most=<reference>`

The reverse of the above.

- `between=<start reference> and <end reference>`

True if `<date>` lies between the two reference dates. If the year is omitted from the reference, only the month and day in `<date>` form part of the test.

- `day of month=<number>`

True if the day of the month in `<date>` is equal to `<number>`.

7.2 The *pgfcalendar* Package Utility Commands

- `end of month=<number>`

This is the reverse of the above in the sense that it's testing `<number>` against the day of the month counting backward from the end of the month. So `end of month=1` yields true if `<date>` is the last day of the month, and `end of month=2` yields true if `<date>` is the penultimate day of the month. If `<number>` is omitted, it's assumed to be 1.

EXAMPLE:

```
2014-03-20 is in the
\pgfcalendarifdate{2014-03-20}{at most=06-last}
{first}%
{second}%
\space half of the year.
```

↑ Input

↓ Input

produces:

7.2 The *pgfcalendar* Package Utility Commands

2014-03-20 is in the first half of the year.

Output

What happens if your date isn't in the form `<yyyy>-<m>-<d>`? For example, it might be in the form `<m>/<d>/<yyyy>`. Recall from §2.1.1 that the `\def` primitive can be used to define a macro that has a custom syntax. It's therefore possible to define a command that will parse this syntax and convert it:

```
\def\parsemdydate#1/#2/#3\endparsemdydate{#3-#1-#2}
```

Input

(The `\endparsemdydate` token is just an end placeholder, not a command that needs defining.)

Now

```
\parsemdydate 3/19/2014\endparsemdydate
```

Input

expands to 2014-3-19. Remember that if the date is stored in a macro, for example:

```
\newcommand*{\mydate}{3/19/2014}
```

Input

then you first need to expand the macro before it can be parsed by `\parsemdydate`. (Recall `\expandafter` from §2.7.2.)

```
\expandafter\parsemdydate\mydate\endparsemdydate
```

Input

EXAMPLE 38. CALCULATING AGES

Remember that the sample `people.csv` file and `people` SQL table included a date of birth field (labelled `dob`). This example computes the ages of each person in that data. This is done by first computing the Julian day number for today. Then for each person in the database, the Julian day number is computed for that person's date of birth. This number is subtracted from the Julian day number for today. This gives the total number of days since that person was born. This number is then divided by 356 to give an approximate age in years. (Recall TeX's integer arithmetic described in §2.1.3.)

↑ Input

```
\documentclass[captions=tableheading]{scrartcl}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage{pgfkeys,pgfcalendar}
\usepackage{datatool}

\DTLloaddb{people}{people.csv}
```

7.2 The *pgfcalendar* Package Utility Commands

7.2 The *pgfcalendar* Package Utility Commands

```
% Compute the Julian day number for the date of birth  
\pgfcalendardatejulian{\DoB}{\julianday} %  
% Compute \age = (\juliantoday - \julianday)/365  
\age=\juliantoday  
\advance\age by -\julianday  
\divide\age by 365  
\number\age  
}%  
\end{tabular}  
\end{table}  
  
\end{document}
```

↓ Input

This produces Table 7.1 and the text:

Ages as of 2015-9-30 are listed in Table 7.1.

Output

You can [download](#) or [view](#) this example. Remember that if your dates are in a different numerical format, for example, $\langle m \rangle / \langle d \rangle / \langle yyyy \rangle$, you need to convert them as described above. For example, replace

```
\pgfcalendardatejulian{\DoB}{\julianday}
```

Input

with

7.2 The *pgfcalendar* Package Utility Commands

Output

Table 7.1 Ages

Name	Age
Polly Parrot	44
Mabel Canary	47
Zöe Zebra	26
José Arara	24
Dickie Duck	62
Fred Canary	48

```
\pgfcalendardatejulian
  {\expandafter\parsemdydate\DoB\endparsemdydate}{\julianday}
```

↑ Input

↓ Input

where `\parsemdydate` is as described above.

7.3 □ Displaying a Date

In addition to the utility commands described [above](#), the `pgfcalendar` package also provides commands to display month names and the day of week names.

`\pgfcalendarweekdayname{<week day number>}`

[Definition](#)

This expands to a textual representation of the day of the week, where the numbering starts from 0 (Monday).

`\pgfcalendarweekdayshortname{<week day number>}`

[Definition](#)

This expands to an abbreviated textual representation of the day of the week, ("Mon", "Tue", etc) where the numbering starts from 0 (Monday).

`\pgfcalendarmonthname{<month number>}`

[Definition](#)

This expands to a textual representation of the month name.

`\pgfcalendarmonthshortname{<month number>}`

[Definition](#)

This expands to an abbreviated textual representation of the month name ("Jan", "Feb", etc).

7.3 Displaying a Date

If you want the name in another language, you need to load the translator package¹ as well as babel [7]. For example:

```
\documentclass[french,german]{article}  
  
\usepackage{babel}  
\usepackage{translator}  
\usepackage{pgfkeys,pgfcalendar}
```

↑ Input

↓ Input

(Only a limited number of languages are supported.)

EXAMPLE:

Today's date can be formatted using the month name:

```
\number\day~\pgfcalendarmonthname{\month} \number\year
```

Input

which produces:

¹provided with beamer [103]

7.3 Displaying a Date

30 September 2015

Output

Alternatively, you can define a custom command called, say, `\datefmt` that formats any date:

```
\newcommand*{\datefmt}[3]{%
  \number#3\~\pgfcalendarmonthname{#2} \number#1%
}
```

↑ Input

↓ Input

This has the syntax:

```
\datefmt{<year>}{'<month>'}{'<day>}'
```

Definition

so now today's date can be formatted using:

```
\datefmt{\year}{\month}{\day}
```

Input

Alternatively, a specific date, for example, 2014-01-31 can be formatted using:

```
\datefmt{2014}{1}{31}
```

Input

which produces:

7.3 Displaying a Date

31 January 2014

Output

If you want to use an ordinal instead of a plain number for the day of the month (for example, 1st instead of 1), then you can use TeX's `\ifcase` conditional:

```
\ifcase <number>
  <case 0 code>%
\or
  <case 1 code>%
\or
  <case 2 code>%
\or
  ...
\else
  <default code>%
\fi
```

Definition

This tests `<number>`. If `<number>` equals 0, `<case 0 code>` is performed. If `<number>` equals 1, `<case 1 code>` is performed. If `<number>` equals 2, `<case 2 code>` is performed, etc. If none of the cases match, `<default code>` is performed. (The `\else <default code>` part may be omitted.) Since there are a maximum of 31 days in a month, 32 cases (including the unnecessary case 0) are needed:

7.3 Displaying a Date

```
\newcommand*{\ord}[1]{%
  \number#1%
  \ifcase#1\or st\or nd\or rd\or th\or th\or th\or th\or
  th\or th\or th\or th\or th\or th\or th\or th\or th\or
  th\or th\or th\or th\or th\or th\or th\or th\or th\or
  th\or th\or th\or th\or th\or th\or th\or th\or th\fi
}
```

↑ Input

Now

```
\ord{1}
```

Input

produces:

1st

Output

and

```
\ord{2}
```

Input

produces:

7.3 Displaying a Date

2nd

Output

etc. So the definition of \datefmt can now be defined as:

```
\newcommand*{\datefmt}[3]{%
  \ord{#3}~\pgfcalendarmonthname{#2} \number#1%
}
```

↑ Input

↓ Input

If you like, you can redefine \today to use this format:

```
\renewcommand*{\today}{\datefmt{\year}{\month}{\day}}
```

Input

 Take care if you use babel as it redefines \today every time the language changes (including at the beginning of the document environment). It does this via the commands \date<language> that are invoked when the current language switches to <language>. Each \date<language> command redefines \today to use the format for <language>. So if you want to redefine \today when you are using babel, you need to redefine \date<language>.

7.3 Displaying a Date

EXAMPLE:

Suppose I'm using babel with the british option. In this case, the date is reset using \datebritish so I need to redefine it to use my own format instead:

```
\renewcommand*\datebritish{%
    \renewcommand*\today{\datefmt{\year}{\month}{\day}}%
}
```

↑ Input

↓ Input

EXAMPLE 39. CUSTOM DATE FORMATTING

Suppose now you want to include the day of the week in your custom date format, or perhaps you want to be able to specify the date in the ISO numeric format, with possibly an increment, as with the first argument of \pgfcalendar.datetojulian.

Recall from §7.2 that the week day can be obtained from a Julian day number using \pgfcalendar.juliantoweekday and the Julian day number can be obtained from a date using \pgfcalendar.datetojulian. In this example, I'm going to define a new command called \printdate with the syntax:

7.3 Displaying a Date

```
\printdate{<date>}
```

Definition

This will display the date in the form: *<day name> <day of month number> <month name> <year>*. First, two new count registers need to be defined:

```
\newcount\julianday  
\newcount\dayofweek
```

↑ Input

↓ Input

Next define the new command:

```
\newcommand*\printdate[1]{%  
  \pgfcalendardatejulian{\#1}{\julianday}-%  
  \pgfcalendarjuliantodate{\julianday}{\thisyear}{\thismonth}  
  {\thisday}-%  
  \pgfcalendarjuliantoweekday{\julianday}{\dayofweek}-%  
  % Now display the date:  
  \datefmt[\dayofweek]{\thisyear}{\thismonth}{\thisday}-%  
}
```

↑ Input

↓ Input

7.3 Displaying a Date

The actual date format, including the day of week, is dealt with by a new version of `\datefmt` that now has four arguments:

`\datefmt[<day of week>]{<yyyy>}{<m>}{<d>}`

Definition

This command is defined as follows:

```
\newcommand*{\datefmt}[4][]{%
  \ifstrempty{#1}%
  {}% day of week missing
  {%
    \pgfcalendarweekdayname{#1}\space
  }%
  \ord{#4}~\pgfcalendarmonthname{#3} \number#2%
}
```

↑ Input

↓ Input

This uses etoolbox's `\ifstrempty` command to omit the day of week name if the optional argument is absent. This means that you can still directly use, for example:

7.3 Displaying a Date

```
\datefmt{2014}{1}{31}
```

Input

and not worry about the week day.

Now this new `\printdate` command can be used in the document. For example:

```
\printdate{2014-05-last}
```

Input

produces:

Saturday 31st May 2014

Output

If you want `\today` to use the same format, then you can just redefine `\today`:

```
\renewcommand*\today{%
\pgfcalendardatejulian{\year-\month-\day}{\julianday}%
\pgfcalendarjuliantoweekday{\julianday}{\dayofweek}%
\datefmt[\dayofweek]{\year}{\month}{\day}%
}
```

↑ Input

↓ Input

7.3 Displaying a Date

Again, you need to put this in the definition of `\date<language>` if you are using babel.

You can [download](#) or [view](#) a complete document.

If this is a format you are likely to use in multiple documents, you might want to define your own custom package called, say, `mycustomdate`. This requires creating a file called `mycustomdate.sty`, that contains the following:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{mycustomdate}[2014/03/19 1.0 My custom date
format]

\RequirePackage{etoolbox}
\RequirePackage{pgfkeys,pgfcalendar}

% Command definitions for \printdate, \today, \datefmt
% \ord (if required) and \date<language> (if required).

\endinput
```

↑ Input

↓ Input

7.3 Displaying a Date

This file should then be saved in your TEXMF path. For example, if you are using a Unix-like operating system, you can save it in, say, `~/texmf/tex/latex/mystuff/` (see [Volume 1 \[93, §A\]](#)).

If you're unfamiliar with writing packages, here's a brief explanation of the commands used above:

`\NeedsTeXFormat{<format>}[<version>]`

[Definition](#)

This should be the first statement of any class or package and is used to identify the TeX format and, optionally, the version date. For a L^AT_EX 2 ϵ class or package, the `<format>` should be L^AT_EX2e. (Other formats may not define this command.) The version date, if present, must be in the numeric form `<yyyy>/<mm>/<dd>` (two digits are required for both the month and day numbers).

`\ProvidesPackage{<name>}[<version>]`

[Definition](#)

This command identifies the package name and optionally a version. The `<name>` should match the filename (without the extension), so a package called `mystuff` should be in a file called `mystuff.sty`. The `<version>` should start with a numeric date in the form `<yyyy>/<mm>/<dd>` and may optionally be followed by a version number and a brief description.

7.3 Displaying a Date

`\RequirePackage[⟨options⟩]{⟨name⟩}[⟨version⟩]`

Definition

This is analogous to `\usepackage` but is for use in a class or package. The final optional argument `⟨version⟩` indicates that the package must be at least that version. If an older version is installed a warning is issued.

`\endinput`

Definition

This is a `TEX primitive` that instructs `TEX` to stop reading the current file. Anything following this command is skipped. (Some packages have their documentation in the `.sty` file after `\endinput`, but this practice has been deprecated in favour of providing the documentation as a PDF.)

Once you have added `mycustomdate.sty` to your `TEX` path, you can now load this package in your document via

`\usepackage{mycustomdate}`

Input

Take care if you need to use `babel`. If this custom package includes code to redefine `\date⟨language⟩` you will need to load `babel` first (and remember to load the `translator` package as well for the month and day of week names). Alternatively, you can check for the existence of `\date⟨language⟩` at the start of the `document` environment, and redefine it if it exists:

↑ Input

7.3 Displaying a Date

```
\AtBeginDocument{%
  \ifdef{\datebritish}{% check if \datebritish exists
    {%
      it does exist, so redefine it
      \renewcommand*\datebritish{%
        \renewcommand*\today{%
          \pgfcalendardatejulian{\year-\month-\day}{\julianday}%
          \pgfcalendarjuliantoweekday{\julianday}{\dayofweek}%
          \datefmt[\dayofweek]{\year}{\month}{\day}%
        }%
      }%
      \datebritish
    }%
  }%
  {}% doesn't exist, do nothing
}
```

↓ Input

EXAMPLE 40. CUSTOM DATE PACKAGE

Putting the above together, here's a complete example that defines a package that uses a British date format:

```
\NeedsTeXFormat{LaTeX2e}
```

↑ Input

7.3 Displaying a Date

```
\ProvidesPackage{mycustomdate}[2014/03/19 1.0 My custom date
format]

\RequirePackage{etoolbox}%
\RequirePackage{pgfkeys,pgfcalendar}

% Define an ordinal command:
\newcommand*\ord[1]{%
  \number#1%
  \ifcase#1\or st\or nd\or rd\or th\or th\or th\or th\or
  th\or th\or th\or th\or th\or th\or th\or th\or th\or
  th\or th\or th\or th\or st\or nd\or rd\or th\or th\or
  th\or th\or th\or th\or th\or st\fi
}

% Define registers needed by \printdate:
\newcount\julianday
\newcount\dayofweek

% Define generic date format:
\newcommand*\datefmt[4][]{%
  \ifstrempty{#1}{%
```

7.3 Displaying a Date

```
{ }% day of week missing
{%
  \pgfcalendarweekdayname{\#1}\space
}%
\ord{\#4}~\pgfcalendarmonthname{\#3} \number{\#2}
}

% Define command to read ISO date and then use \datefmt
\newcommand*{\printdate}[1]{%
  \pgfcalendardatejulian{\#1}{\julianday}%
  \pgfcalendarjuliantodate{\julianday}{\thisyear}{\thismonth}{\thisday}%
  \pgfcalendarjuliantoweekday{\julianday}{\dayofweek}%
  % Now display the date:
  \datefmt[\dayofweek]{\thisyear}{\thismonth}{\thisday}%
}

% Redefine \today to use the same format:
\renewcommand*{\today}{%
  \pgfcalendardatejulian{\year-\month-\day}{\julianday}%
  \pgfcalendarjuliantoweekday{\julianday}{\dayofweek}%
  \datefmt[\dayofweek]{\year}{\month}{\day}%
}
```

7.3 Displaying a Date

```
% Check if babel is used with the british option:  
\AtBeginDocument{  
  \ifdef{\datebritish}{% check if \datebritish exists  
    {  
      it does exist, so redefine it  
      \renewcommand*\datebritish{  
        \renewcommand*\today{}%  
        \pgfcalendardatejulian{\year-\month-\day}{\julianday}%  
        \pgfcalendarjuliantoweekday{\julianday}{\dayofweek}%  
        \datefmt[\dayofweek]{\year}{\month}{\day}%  
      }%  
    }%  
    \datebritish  
  }%  
  {}% doesn't exist, do nothing  
}  
  
\endinput
```

↓ Input

(You can [download](#) this package.)

And here's a document that uses this package:

↑ Input

7.3 Displaying a Date

```
\documentclass{article}  
  
\usepackage{mycustomdate}  
  
\begin{document}
```

Today: \today.

Tomorrow: \printdate{\year-\month-\day+1}.

Yesterday: \printdate{\year-\month-\day+-1}.

The first day of this month: \printdate{\year-\month-1}.

The last day of this month: \printdate{\year-\month-last}.

A specific date: \printdate{2014-3-20}.

A date without the day of week: \datefmt{2014}{3}{20}.

```
\end{document}
```

↓ Input

You can [download](#) or [view](#) this document.

7.4 Parsing and Displaying Times

The pgfcalendar doesn't provide any time-related utilities. TeX's `\time primitive` expands to the current time in terms of the number of minutes since midnight. The datetime package works out the current hour and minute by performing some arithmetic on the value of `\time`, but since `\time` is an integer number of minutes, there's no information about the number of seconds nor is there any information about the time zone. (The new datetime2 package uses the methods described in this section to determine the current time.) PDFTeX comes with a `primitive2` called

`\pdfcreationdate`

Definition

that expands to `D:<YYYY><MM><DD><hh><mm><ss><time zone>` where `<YYYY>` is the year (four digits), `<MM>` is the month number (two digits), `<DD>` is the day of the month (two digits), `<hh>` is the hour (two digits), `<mm>` is the number of minutes past the hour (two digits), `<ss>` is the number of seconds past the minute (two digits) and `<time zone>` is the time zone, which may be `Z` (for UTC+00) or `+<HH>'<mm>'` (for UTC+`<HH>:<mm>`) or `-<HH>'<mm>'` (for UTC−`<HH>:<mm>`).

The value of `\pdfcreationdate` is set at the start of the PDFTeX run.

²For further details about PDFTeX `primitives` see the PDFTeX documentation [106].

7.4 Parsing and Displaying Times

EXAMPLE:

\pdfcreationdate

Input

produces:

D:20150930082945+01'00'

Output

Recall the \parsemdydate command defined in §7.2 used \def to parse a date string. A similar method can be employed here, but unfortunately it's more complicated. At first glance it looks as though we can define a command in the form:

\def\parsepdfdatetime D:#1\endparsepdfdatetime{(code)}

X

However if we try this out (ignoring the argument for the time being):

```
\def\parsepdfdatetime D:#1\endparsepdfdatetime{}  
\expandafter\parsepdfdatetime\pdfcreationdate\endparsepdfdatetime
```

↑ Input

↓ Input

7.4 Parsing and Displaying Times

we get an error:

```
! Use of \parsepdfdatetime doesn't match its definition.  
<inserted text> D  
:20140319185833Z  
1.10 \expandafter\parsepdfdatetime\pdfcreationdate  
                                \endparsepdfdatetime
```

The problem is the initial D as the following works fine:³

```
\def\parsepdfdatetime#1:#2\endparsepdfdatetime{}  
\expandafter\parsepdfdatetime\pdfcreationdate\endparsepdfdatetime
```

The first argument (#1) will always be D and can be ignored. Since \TeX only allows a maximum of nine arguments, this leaves eight arguments left, which can pick up the year (#2#3#4#5), month (#6#7) and day (#8#9) digits. This information is already available from \TeX 's `\year`, `\month` and `\day` primitives, however PDFT \TeX provides a similar primitive:

³It's the category code of the character "D" in the expansion of `\pdfcreationdate` that's the problem. If it's first changed to "other" (category code 12) before defining `\parsepdfdatetime` then the error won't occur.

7.4 Parsing and Displaying Times

\pdffilemoddate{\langle filename\rangle}

Definition

that expands to the modification date and time for the file given by `\langle filename\rangle`, and this uses the same format, so `\parsepdfdatetime` should still save the year, month and day information to be more generally useful.

In order to get around the nine argument maximum `\parsepdfdatetime` needs to call another command that will parse the remainder:

```
\def\parsepdfdatetime#1:#2#3#4#5#6#7#8#9{%
  \def\theyear{#2#3#4#5}%
  \def\themonth{#6#7}%
  \def\theday{#8#9}%
  \parsepdftime
}
```

↑ Input

↓ Input

Note that the end placeholder `token` `\endparsepdfdatetime` is no longer in the argument syntax of `\parsepdfdatetime`. It's now in the argument syntax of the new `\parsepdftime` command, which picks up the remaining time information:

7.4 Parsing and Displaying Times

↑ Input

```
\def\parsepdfdate{\#1\#2\#3\#4\#5\#6\#7}\endparsepdfdate{%
  \def\thehour{\#1\#2}%
  \def\theminute{\#3\#4}%
  \def\thesecond{\#5\#6}%
  \def\thetimezone{\#7}%
}
```

↓ Input

The hour digits are now given by #1#2, the minute digits are #3#4 the second digits are #5#6 and the time zone information is in the final argument #7. This information has been stored in the new commands \thehour, \theminute, \thesecond and \thetimezone. If you want \thetimezone to be in the format <sign><HH>:<mm> (where <sign> is either + or -) then replace:

```
\def\thetimezone{\#7}%

```

Input

with

↑ Input

```
\ifstreq{\#7}{Z}{
```

7.4 Parsing and Displaying Times

```
{%
  \def\thetimezone{+00:00}%
}%
{%
  \parsepdftimezone#7%
}%

```

↓ Input

where `\parsepdftimezone` is defined as:

```
\def\parsepdftimezone#1'#2'{%
  \def\thetimezone{-#1:#2}%
}
```

↑ Input

(`\ifstrequal` is defined by etoolbox and tests if two strings are equal, but unlike `\ifthenelse{\equal{<string1>}{<string2>}}{<true>}{{<false>}}`, `\ifstrequal` doesn't perform any expansion on the strings.)

As with the `\datefmt` command defined in the previous section, we can also define an analogous command to format the time:

7.4 Parsing and Displaying Times

```
\newcommand*{\timefmt}[4]{%
  #1:#2:#3#4%
}
```

↑ Input

This has the syntax:

```
\timefmt{<hour>}{'<minutes>}{'<seconds>}{'<UTC offset>}'
```

Definition

For example:

```
\timefmt{11}{03}{01}{+01:00}
```

Input

produces:

```
11:03:01+01:00
```

Output

Another possible definition is:

```
\newcommand*{\timefmt}[4]{%
  #1:#2%
  \ifstrempy{#3}{}{ test for empty 3rd argument}
```

↑ Input

7.4 Parsing and Displaying Times

```
{}% no seconds specified  
{:#3}% seconds  
\ifstrempty{#4}% test for empty 4th argument  
{}% no time zone  
{#4}%  
}  
↓ Input
```

This uses etoolbox's `\ifstrempty` command to determine whether or not `(seconds)` or `(UTC offset)` have been specified. Alternatively, the time zone information can be dealt with by another command called, say, `\timezonefmt`:

```
\newcommand*{\timefmt}[4]{%  
#1:#2%  
\ifstrempty{#3}% test for empty 3rd argument  
{}% no seconds specified  
{:#3}% seconds  
\timezonefmt{#4}% time zone  
}  
↑ Input
```

7.4 Parsing and Displaying Times

Since it's possible that the time zone might not be fully expanded (for example, the argument might be `\thetimezone`), the new `\timezonefmt` first fully expands its argument before parsing it:

```
\newcommand*{\timezonefmt}[1]{%
  \edef\thistimezone{#1}%
  \ifdefempty{\thistimezone}%
  {}% empty argument
  {%
    \expandafter\@timezonefmt\thistimezone\@endtimezonefmt
  }%
}
```

↑ Input

↓ Input

Now the actual parsing is done by a new **internal command** with the syntax:

`\@timezonefmt<HH>:<mm>\@endtimezonefmt`

Definition

Here's one possible definition of `\@timezonefmt` that just displays "Z" if both `<HH>` and `<mm>` are zero, otherwise it either does just `<HH>` if `<mm>` is zero or it does `<HH>:<mm>`

7.4 Parsing and Displaying Times

↑ Input

```
\def\@timezonefmt#1:#2\@endtimezonefmt{%
  \ifnum#2=0\relax
    \ifnum#1=0\relax
      Z%
    \else
      #1%
    \fi
  \else
    #1:#2%
  \fi
}
```

↓ Input

EXAMPLE:

Since `\pdfcreationdate` is set at the start of the L^AT_EX run, you only need to parse it once:

↑ Input

```
\expandafter\parsepdfdatetime\pdfcreationdate\endparsepdfdatetime
\let\pdfhour\thehour
```

7.4 Parsing and Displaying Times

```
\let\pdfminute\theminute
\let\pdfsecond\thesecond
\let\pdftimezone\thetimezone
\newcommand*{\pdfnowtime}{%
  \timefmt{\pdfhour}{\pdfminute}{\pdfsecond}{\pdftimezone}}
```

↓ Input

The time stamp for the L^AT_EX run can now be inserted into your document using this new `\pdfnowtime` command:

This PDF was created at `\pdfnowtime`.

Input

produces:

This PDF was created at 08:29:45+01.

Output

EXAMPLE 41. CUSTOM DATE AND TIME PACKAGE

This example extends the custom package described in [Example 40](#). I've added the L^AT_EX internal command:

```
\two@digits{<number>}
```

Definition

which ensures `<number>` has at least two digits. Take care when using commands that print numbers, such as `\two@digits` or `\number`, as you can

7.4 Parsing and Displaying Times

unexpectedly lose following spaces. It's for this reason that I've occasionally used `\relax` or `_` (backslash space) in the code below.

This example package is now called `mycustomdatetime` so it needs to have the filename `mycustomdatetime.sty` and the package declaration should be modified accordingly:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{mycustomdatetime}[2014/03/20 1.0 My custom date
and time format]
```

↑ Input

Remember the `etoolbox` package is required:

```
\RequirePackage{etoolbox}
```

Input

Now the user command

```
\timefmt{\langle hh \rangle}{\langle mm \rangle}{\langle ss \rangle}{\langle HH \rangle}:{\langle SS \rangle}
```

Definition

is defined:

7.4 Parsing and Displaying Times

↑ Input

```
\newcommand*{\timefmt}[4]{%
  \two@digits{#1}:\two@digits{#2}%
  \ifstrempty{#3}{% test for empty 3rd argument
    {}% no seconds specified
    {:\two@digits{#3}}% seconds
  \timezonefmt{#4}% time zone
  \relax
}
```

↓ Input

and its helper time zone formatting command:

```
\timezonefmt{<HH>:<SS>}
```

Definition

is defined:

↑ Input

```
\newcommand*{\timezonefmt}[1]{%
  \edef\thistimezone{#1}%
  \ifdefempty{\thistimezone}{%
    {}% empty argument
  }
```

7.4 Parsing and Displaying Times

```
{%
  \expandafter\@timezonefmt\thistimezone\@endtimezonefmt
}%
}
```

↓ Input

along with its [internal command](#):

```
\def\@timezonefmt#1:#2\@endtimezonefmt{%
  \ifnum#2=0\relax
    \ifnum#1=0\relax
      Z%
    \else
      #1%
    \fi
  \else
    #1:#2%
  \fi
}
```

↑ Input

7.4 Parsing and Displaying Times

If you want to use `\two@digits` in the time zone, you need to be careful with the plus or minus sign:

```
\def\@timezonefmt#1:#2\@endtimezonefmt{%
  \ifnum #2=0\relax
    \ifnum #1=0\relax
      Z%
    \else
      \ifnum #1<0 $-$\two@digits{-#1}\else +\two@digits{#1}\fi
    \fi
  \else
    \ifnum #1<0 $-$\two@digits{-#1}\else +\two@digits{#1}\fi
    :\two@digits{#2}%
  \fi
}
}
```

↑ Input

↓ Input

(The minus sign has been placed in math-mode using `$-$` to ensure it's displayed as a real minus sign rather than as a hyphen. If for some reason you need to use `\timefmt` in math-mode, I suggest you put it in inside the argument of amsmath's `\text` command [1].)

7.4 Parsing and Displaying Times

Now for the commands that can parse the PDF date format:

↑ Input

```
\def\parsepdfdatetime#1:#2#3#4#5#6#7#8#9{%
  \def\theyear{#2#3#4#5}%
  \def\themonth{#6#7}%
  \def\theday{#8#9}%
  \parsepdftime
}

\def\parsepdftime#1#2#3#4#5#6#7\endparsepdfdatetime{%
  \def\thehour{#1#2}%
  \def\theminute{#3#4}%
  \def\thesecond{#5#6}%
  \ifstrequal{#7}{Z}
  {%
    \def\thetimezone{+00:00}%
  }%
  {%
    \parsepdftimezone#7%
  }%
}
```

7.4 Parsing and Displaying Times

```
\def\parsepdftimezone#1'#2'{%
  \def\thetimezone{#1:#2}%
}
```

↓ Input

Provide a convenient way of displaying the document build time:

```
\expandafter\parsepdfdatetime\pdfcreationdate\endparsepdfdatetime
\let\pdfhour\thehour
\let\pdfminute\theminute
\let\pdfsecond\thesecond
\let\pdftimezone\thetimezone

\newcommand*\pdfnowtime{%
  \timefmt{\pdfhour}{\pdfminute}{\pdfsecond}{\pdftimezone}}
```

↑ Input

and for a complete date and time stamp:

7.4 Parsing and Displaying Times

```
\newcommand*{\pdfnow}{\today\_\pdfnowtime}
```

Input

Alternatively if you want a numeric date independent of the definition of `\today`:

```
\newcommand*{\pdfnow}{%
\year-\two@digits{\month}-\two@digits{\day}\space
\pdfnowtime
}
```

↑ Input

↓ Input

Similarly define a command with the syntax:

```
\filedate{<filename>}
```

Definition

that will display the time stamp of a file:

```
\newcommand*{\filedate}[1]{%
\expandafter\parsepdfdatetime\pdffilemoddate{#1}\endparsepdfdatetime
\datefmt{\theyear\themonth\theday}\space
\timefmt{\thehour\theminute\thesecond\thetimezone}}%
```

↑ Input

↓ Input

7.4 Parsing and Displaying Times

Alternatively, if you want the day of week information to also be shown:

```
\newcommand*{\filedate}[1]{%
  \expandafter\parsepdfdatetime\pdffilemoddate{#1}\endparsepdfdatetime
  \printdate{\theyear-\themonth-\theday}\_
  \timefmt{\thehour}{\theminute}{\thesecond}{\thetimezone}%
}
```

↑ Input

↓ Input

Or if you just want a numeric format regardless of the definition of \printdate:

```
\newcommand*{\filedate}[1]{%
  \expandafter\parsepdfdatetime\pdffilemoddate{#1}\endparsepdfdatetime
  \theyear-\two@digits{\themonth}-\two@digits{\theday}\_
  \timefmt{\thehour}{\theminute}{\thesecond}{\thetimezone}%
}
```

↑ Input

↓ Input

7.4 Parsing and Displaying Times

The rest of the package code is as described in [Example 40](#), including the definitions of `\datefmt` and `\printdate`. (You can [download](#) the complete package.) Here's an example document that uses this new package:

↑ Input

```
\documentclass{article}

\usepackage{mycustomdatetime}

\begin{document}

The file \jobname.tex was last modified on:
\filedate{\jobname.tex}.

The PDF was built by \TeX\ on: \pdfnow.
```

```
Format a specific time (Zulu time):
\timefmt{8}{10}{35}{+0:00}.
```

```
Format a specific time (non-Zulu time):
\timefmt{8}{10}{35}{+1:00} or
\timefmt{8}{10}{35}{-4:30} or
```

7.4 Parsing and Displaying Times

```
\timefmt{8}{10}{35}{+5:45}.
```

Format a specific time without a time zone:

```
\timefmt{8}{10}{35}{}.
```

Format a specific time with a time zone but without seconds:

```
\timefmt{8}{10}{ }{+00:00}.
```

Format a specific time without a time zone or seconds:

```
\timefmt{8}{10}{ }{ }.
```

```
\end{document}
```

↓ Input

You can [download](#) or [view](#) this document.

EXERCISE 22. DISPLAYING TIMES

Add a command to the mycustomdatetime package described in [Example 41](#) that has the syntax:

7.4 Parsing and Displaying Times

```
\printdatetime{\langle YYYY\rangle-\langle MM\rangle-\langle DD\rangle \langle hh\rangle:\langle mm\rangle:\langle ss\rangle\langle UTC offset\rangle}
```

Definition

This command should be equivalent to:

```
\printdate{\langle YYYY\rangle-\langle MM\rangle-\langle DD\rangle} \timefmt{\langle hh\rangle}{\langle mm\rangle}{\langle ss\rangle}{\langle UTC offset\rangle}
```

Input

Example usage:

```
\printdatetime{2014-03-25 01:23:15+00:00}  
\printdatetime{2014-03-24 23:31:58+01:00}  
\printdatetime{2014-03-24 16:28:56-06:00}  
\printdatetime{2014-03-24 14:45:23+08:00}  
\printdatetime{2014-03-25 03:12:04-04:30}  
\printdatetime{2014-03-24 03:45:24-04:30}  
\printdatetime{2014-03-24 21:20:24+05:45}
```

↑ Input

↓ Input

FOR THE MORE ADVENTUROUS:

Suppose I now need all the times in a common time zone for easier comparison. Make a new command called, say, `\printzuludatetime` that has

7.5 Displaying a Calendar

the same syntax as `\printdatetime` but it converts the date and time to Zulu time (UTC+00:00) before displaying it.

You can [download](#) or [view](#) a solution. (The new `datetime2` package [101] now comes with commands that perform a similar conversion in the accompanying `datetime2-calc` package.)

7.5 □ Displaying a Calendar

The `pgfcalendar` package provides the command:

`\pgfcalendar{<prefix>}{{<start date>}}{<end date>}{<code>}`

Definition

This is a loop macro that iterates from `<start date>` to `<end date>` and performs `<code>` at each iteration. Within `<code>` you can access information about the current iteration using:

- `\pgfcalendarcurrentjulian` This is a TeX count register that holds the Julian day number for the current iteration;
- `\pgfcalendarcurrentweekday` The current week day index (0 for Monday, 1 for Tuesday, etc);

7.5 Displaying a Calendar

- `\pgfcalendarcurrentyear` The current year;
- `\pgfcalendarcurrentmonth` The current month (always two digits with a leading zero, if necessary);
- `\pgfcalendarcurrentday` The current day of the month.

In addition, within `<code>` you can also use:

- `\pgfcalendarprefix` The `<prefix>` parameter;
- `\pgfcalendarbeginiso` The `<start date>` in ISO format;
- `\pgfcalendarbeginjulian` The `<start date>` as a Julian day number;
- `\pgfcalendarendiso` The `<end date>` in ISO format;
- `\pgfcalendarendjulian` The `<end date>` as a Julian day number;
- `\ifdate{<tests>}{{<true code>}}{{<false code>}}`

The same as using `\pgfcalendarifdate` for the current date.

7.5 Displaying a Calendar

- `\pgfcalendar suggestedname`

If `<prefix>` is empty, this expands to an empty string, otherwise it expands to `<prefix>-<YYYY>-<MM>-<DD>` so it can be used, for example, as a node name if the calendar is typeset using the `tikzpicture` environment [102].

- `\pgfcalendar shorthand{<kind>}{{<representation>}}`

This will expand to a representation of the current day, month, year or day of week, depending on whether `<kind>` is `d`, `m`, `y` or `w`. The `<representation>` may be one of:

- Numerical representation with no leading zeros;
- = Numerical representation with a leading space for single digit numbers;
- `0` Numerical representation with a leading zero for single digit numbers;
- `t` Textual representation;
- . Abbreviated textual representation.

Typically you would use:

```
\let\%\pgfcalendar shorthand
```

Input

7.5 Displaying a Calendar

before `\pgfcalendar` so that you can simply write, for example, `\%wt` instead of:

```
\pgfcalendarshorthand{w}{t}
```

Input

but make sure you localise the effect of the `\let` by placing it inside a group or environment so that the normal behaviour of `\%` is restored after the calendar has been typeset.

EXAMPLES:

1. To just display the day of the month from 2014-02-26 to 2014-03-15:

```
\pgfcalendar{}{2014-02-26}{2014-03-15}{%
\pgfcalendarcurrentday\_\_}
```

Input

produces:

26 27 28 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

Output

2. To display the date for each day from 2014-03-01 to 2014-03-04:

7.5 Displaying a Calendar

↑ Input

```
% localise effect of \let  
 \let\%{\pgfcalendarshorthand  
 \pgfcalendar{}{2014-03-01}{2014-03-04}{\%w.  
 \%d- \%mt \%y0\par}  
 }
```

↓ Input

produces:

↑ Output

```
Sat 1 March 2014  
Sun 2 March 2014  
Mon 3 March 2014  
Tue 4 March 2014
```

↓ Output

7.5 Displaying a Calendar

The tikz package (part of the pgf bundle) provides a powerful and user-friendly way of drawing images. An in-depth discussion of the tikz package is beyond the scope of this book, but here's a very brief introduction to drawing nodes in a `tikzpicture` environment to help draw a simple calendar. For more detail about tikz, see the pgf user manual [102].

Within the `tikzpicture` environment, you can use

```
\path[<path options>] (<position>) node[<node options>] (<node name>) {<text>};
```

Definition

to draw a node. Alternatively you can use

```
\node[<node options>] at (<position>) (<node name>) {<text>};
```

Definition

The full syntax is more complicated, but the `(<node name>)` is optional, as are the `key=value lists` `<path options>` and `<node options>`. (Spaces before and after the commas and equal signs are ignored.) The full syntax of `(<position>)` is also quite complicated, but here I'll just use the `(x,y)` syntax.

EXAMPLE:

(Don't forget to load the tikz package.)

↑ Input

7.5 Displaying a Calendar

```
\fbox{%
\begin{tikzpicture}
\path (0,0) node {Mon};
\path (1,0) node {Tue};
\path (2,0) node {Wed};
\path (3,0) node {Thu};
\path (4,0) node {Fri};
\path (5,0) node {Sat};
\path (6,0) node {Sun};
\end{tikzpicture}%
}
```

↓ Input

I've used the `\fbox` command (described in [Volume 1 \[93, §4.7.1\]](#)) to put a border around the picture. Fancier borders can be created using `tikz` commands within the `tikzpicture` environment.

The above code produces:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
-----	-----	-----	-----	-----	-----	-----

Output

It's possible to add a `\pgfcalendar` command to this environment and put the node drawing part in the `<code>` argument. Since `\pgfcalendarcurrentweekday` is an integer from 0 (Monday) to 6 (Sunday),

7.5 Displaying a Calendar

it can be used for the $\langle x \rangle$ coordinate. Since tikz uses a right-handed coordinate system, the row below the $\langle y \rangle = 0$ weekday name row displayed above needs to be negative. For example:

↑ Input

```
\fbox{%
\begin{tikzpicture}
% First row
\path (0,0) node {Mon};
\path (1,0) node {Tue};
\path (2,0) node {Wed};
\path (3,0) node {Thu};
\path (4,0) node {Fri};
\path (5,0) node {Sat};
\path (6,0) node {Sun};
% Second row
\pgfcalendar{}{2014-03-01}{2014-03-02}{
\path (\pgfcalendarcurrentweekday,-1)
node {\pgfcalendarcurrentday};
}
\end{tikzpicture}
}
```

7.5 Displaying a Calendar

↓ Input

This produces:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					01	02

Output

A counter is required if more than one week needs to be displayed. For example:

↑ Input

```
% Define a new count register:  
\newcount\rowcount  
% Initialise:  
\mycount = 1\relax  
% Draw the calendar:  
\fbox{  
 \begin{tikzpicture}  
 % header row  
 \path (0,0) node {Mon};
```

7.5 Displaying a Calendar

```
\path (1,0) node {Tue};  
\path (2,0) node {Wed};  
\path (3,0) node {Thu};  
\path (4,0) node {Fri};  
\path (5,0) node {Sat};  
\path (6,0) node {Sun};  
% Now iterate through the month of March:  
\pgfcalendar{}{2014-03-01}{2014-03-31}  
{% Draw node for current day  
\path  
    (\pgfcalendarcurrentweekday,-\rowcount) % coordinate  
    node {\pgfcalendarcurrentday}; % node  
% Increment row count if today is a Sunday:  
\ifdate{Sunday}{\advance\rowcount by 1}{  
}% end of loop  
\end{tikzpicture}  
}  
]  
↓ Input
```

This now produces the image shown in [Figure 7.1](#).

Nodes can have a border and background. These can be specified in the `[<node options>]`. For example:

7.5 Displaying a Calendar

Output

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					01	02
03	04	05	06	07	08	09
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Figure 7.1 Calendar (Days of March)

7.5 Displaying a Calendar

```
\path (0,0) node[rectangle,draw] {Mon};
```

Input

will draw a rectangular border around the node while

```
\path (0,0) node[circle,fill=cyan] {Mon};
```

Input

will give the node a cyan circular background.

EXAMPLE:

The above example can be modified to include borders and backgrounds:

```
% Define a new count register:  
\newcount\rowcount  
% Initialise:  
\mycount = 1\relax  
% Draw the calendar:  
\fbox{  
 \begin{tikzpicture}  
 \path (0,0) node[circle,fill=yellow] {Mon};  
 \path (1,0) node[circle,fill=yellow] {Tue};  
 \path (2,0) node[circle,fill=yellow] {Wed};  
 \path (3,0) node[circle,fill=yellow] {Thu};
```

↑ Input

7.5 Displaying a Calendar

```
\path (4,0) node[circle,fill=yellow] {Fri};  
\path (5,0) node[circle,fill=cyan] {Sat};  
\path (6,0) node[circle,fill=cyan] {Sun};  
\pgfcalendar{}{2014-03-01}{2014-03-31}{  
    % Draw node for current day  
    \path (\pgfcalendarcurrentweekday,-\rowcount)  
        node[rectangle,draw] {\pgfcalendarcurrentday};  
    % Increment row count if today is a Sunday:  
    \ifdate{Sunday}{\advance\rowcount by 1}{  
    }  
\end{tikzpicture}  
}
```

↓ Input

Instead of repeatedly using the same options it's possible to set them within a local scope using the `scope` environment:

```
% Define a new count register:  
\newcount\rowcount  
% Initialise:  
\mycount = 1\relax
```

↑ Input

7.5 Displaying a Calendar

```
% Draw the calendar:  
\fbox{  
  \begin{tikzpicture}  
    \begin{scope}[every node/.style={circle,fill=yellow}]  
      \path (0,0) node {Mon};  
      \path (1,0) node {Tue};  
      \path (2,0) node {Wed};  
      \path (3,0) node {Thu};  
      \path (4,0) node {Fri};  
      \path (5,0) node[fill=cyan] {Sat};  
      \path (6,0) node[fill=cyan] {Sun};  
    \end{scope}  
    \pgfcalendar{}{2014-03-01}{2014-03-31}{  
      % Draw node for current day  
      \path (\pgfcalendarcurrentweekday,-\rowcount)  
        node[rectangle,draw] {\pgfcalendarcurrentday};  
      % Increment row count if today is a Sunday:  
      \ifdate{Sunday}{\advance\rowcount by 1}{}  
    }  
  \end{tikzpicture}  
}
```

↓ Input

7.5 Displaying a Calendar

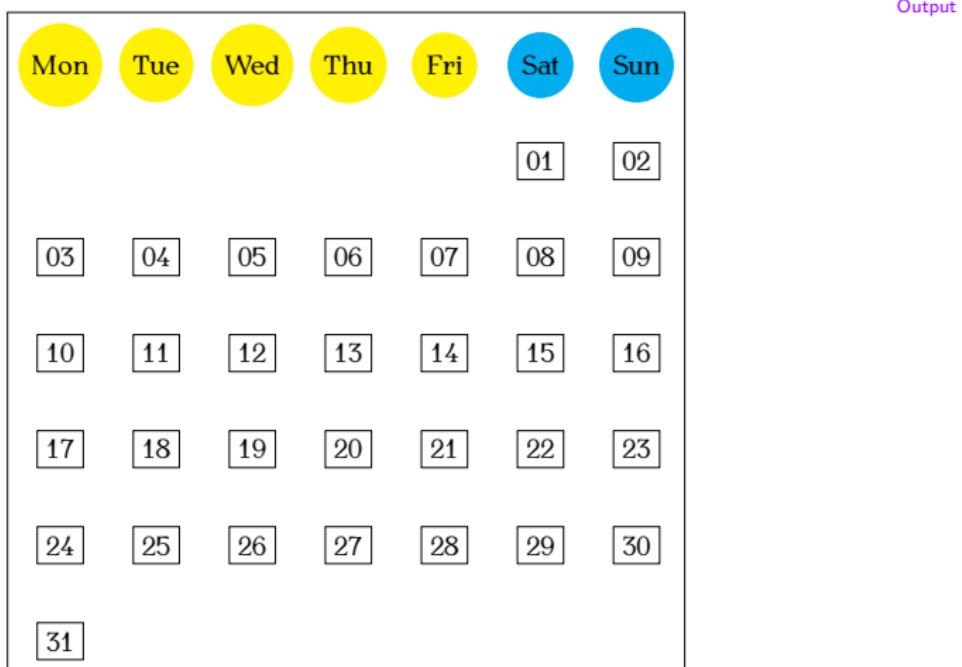


Figure 7.2 Calendar (Node Shapes Added)

7.5 Displaying a Calendar

This produces the image shown in Figure 7.2. The nodes in the first row look a little uneven as the sizes vary according to the node contents. To neaten things up a bit, a minimum size can be imposed on the nodes:

↑ Input

```
% Define a new count register:  
\newcount\rowcount  
% Initialise:  
\mycount = 1\relax  
% Draw the calendar:  
\fbox{  
 \begin{tikzpicture}  
   \begin{scope}[every node/.style={circle,fill=yellow,  
   minimum size=3em}]  
     \path (0,0) node {Mon};  
     \path (1,0) node {Tue};  
     \path (2,0) node {Wed};  
     \path (3,0) node {Thu};  
     \path (4,0) node {Fri};  
     \path (5,0) node[fill=cyan] {Sat};  
     \path (6,0) node[fill=cyan] {Sun};  
   \end{scope}  
 \end{tikzpicture}  
}
```

7.5 Displaying a Calendar

```
\pgfcalendar{}{2014-03-01}{2014-03-31}%
% Draw node for current day
\path (\pgfcalendarcurrentweekday,-\rowcount)
    node[rectangle,draw] {\pgfcalendarcurrentday};
% Increment row count if today is a Sunday:
\ifdate{Sunday}{\advance\rowcount by 1}{}
}
\end{tikzpicture}
}
```

↓ Input

This produces the image shown in [Figure 7.3](#).

However now the nodes are bumping into each other, so they need to be moved apart. The default $\langle x \rangle$ and $\langle y \rangle$ coordinate units are 1 cm. This can be changed in the optional argument of the `tikzpicture` environment. For example:

```
% Define a new count register:
\newcount\rowcount
% Initialise:
\mycount = 1\relax
```

↑ Input

7.5 Displaying a Calendar

Output

Mon	Tue	Wed	Thu	Fri	Sat	Sun
						01
03	04	05	06	07	08	09
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
						31

Figure 7.3 Calendar (Minimum Width Set on Nodes)

7.5 Displaying a Calendar

```
% Draw the calendar:  
\fbox{  
  \begin{tikzpicture}[x=1.5cm,y=1.25cm]  
    \begin{scope}[every node/.style={circle,fill=yellow,  
minimum size=3em}]  
      \path (0,0) node {Mon};  
      \path (1,0) node {Tue};  
      \path (2,0) node {Wed};  
      \path (3,0) node {Thu};  
      \path (4,0) node {Fri};  
      \path (5,0) node[fill=cyan] {Sat};  
      \path (6,0) node[fill=cyan] {Sun};  
    \end{scope}  
    \pgfcalendar{}{2014-03-01}{2014-03-31}{  
      % Draw node for current day  
      \path (\pgfcalendarcurrentweekday,-\rowcount)  
        node[rectangle,draw,minimum width=1cm]  
        {\pgfcalendarcurrentday};  
      % Increment row count if today is a Sunday:  
      \ifdate{Sunday}{\advance\rowcount by 1}{}  
    }  
  \end{tikzpicture}}%
```

7.5 Displaying a Calendar

```
}
```

↓ Input

This produces the calendar shown in [Figure 7.4](#).

The circle and rectangle shapes are always available, but there are other shapes as well that can be loaded via the relevant tikz library, which can be loaded in the preamble using:

```
\usetikzlibrary{<name>}
```

Definition

where `<name>` is the library name. For example, there are some multi-part shapes defined in the `shapes.multipart` library. In order to use these shapes, you not only need

```
\usepackage{tikz}
```

Input

in the preamble but also

```
\usetikzlibrary{shapes.multipart}
```

Input

EXAMPLE:

A rectangular split node with 2 splits can be created using:

7.5 Displaying a Calendar

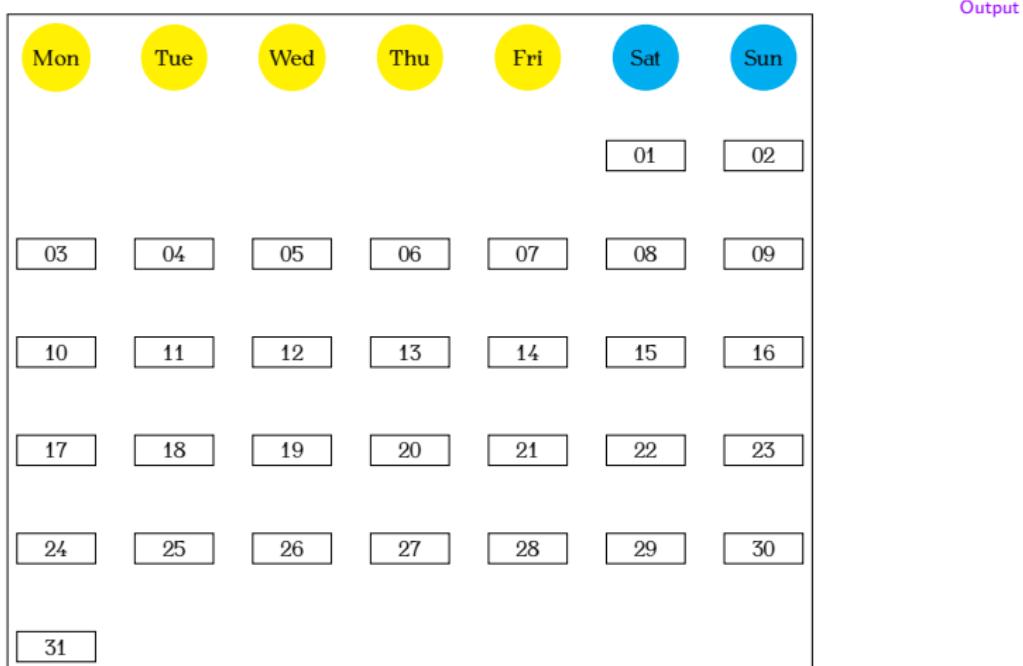


Figure 7.4 Calendar Using Circular and Rectangular Nodes (March 2014)

7.5 Displaying a Calendar

↑ Input

```
\begin{tikzpicture}
\path (0,0)
node[rectangle split,rectangle split parts=2,draw]
{%
    Top
    \nodepart{two}
    Bottom
};
\end{tikzpicture}
```

↓ Input

The `\nodepart{<part>}` command moves from the current split part to the split part identified by `<part>`. In the case of a rectangular split node, the second part is identified by the keyword `two`. The above code produces:



Output

With a vertical split node, such as in the above example, you can set a minimum width using the `minimum width` key, but you can't specify a minimum height. You can, however, specify a height for empty parts using the option `rectangle split empty part height=<length>`. For example:

7.5 Displaying a Calendar

↑ Input

```
\begin{tikzpicture}
  \path (0,0)
    node [
      rectangle split,
      rectangle split parts=2,
      rectangle split empty part height=1cm,
      minimum width=2cm,
      draw
    ]
    {%
      Top
      \nodepart{two}
      % empty bottom part
    };
\end{tikzpicture}
```

↓ Input

This produces:

7.5 Displaying a Calendar



Output

You can specify fill colours for each part using the `rectangle split part fill={⟨colour list⟩}` option, where ⟨colour list⟩ is a comma-separated list of colours for each part, in order. For example:

↑ Input

```
\begin{tikzpicture}
  \path (0,0)
    node
    [
      rectangle split,
      rectangle split parts=2,
      rectangle split empty part height=1cm,
      rectangle split part fill={cyan,magenta},
      minimum width=2cm,
      draw
    ]

```

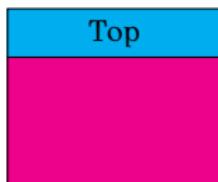
7.5 Displaying a Calendar

```
{%
  Top
  \nodepart{two}
  % empty bottom part
};

\end{tikzpicture}
```

↓ Input

This produces:



Output

Since tikz loads the xcolor package [41], you can specify colours using the xcolor syntax. For example:

```
\begin{tikzpicture}
  \path (0,0)
  node
```

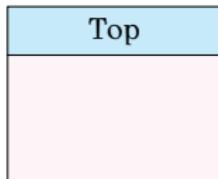
↑ Input

7.5 Displaying a Calendar

```
[  
    rectangle split,  
    rectangle split parts=2,  
    rectangle split empty part height=1cm,  
    rectangle split part fill={cyan!20,magenta!5},  
    minimum width=2cm,  
    draw  
]  
{%  
Top  
\nodepart{two}  
% empty bottom part  
};  
\end{tikzpicture}
```

↓ Input

This produces:



Output

7.5 Displaying a Calendar

Now the fill colour for the top part is 20% cyan tint and the fill colour for the bottom part is 5% magenta tint. This isn't a great colour scheme, but it's just used for illustrative purposes.

EXAMPLE 42. CALENDAR FOR MAY 2014

The above can be put together to create a calendar for the month of May 2014:

↑ Input

```
\newcount\rowcount
\rowcount=1\relax

\fbox{%
\let\%\pgfcalendarshorthand
\begin{tikzpicture}[x=1.5cm,y=1.75cm]
\begin{scope}
[every node/.style={rectangle,fill=green!5,minimum width=1.4cm}]
\path (0,0) node {Mon};
\path (1,0) node {Tue};
\path (2,0) node {Wed};
\path (3,0) node {Thu};
\path (4,0) node {Fri};
```

7.5 Displaying a Calendar

```
\path (5,0) node {Sat};  
\path (6,0) node {Sun};  
\end{scope}  
\pgfcalendar[]{2014-05-01}{2014-05-31}  
{  
    \path (\pgfcalendarcurrentweekday,-\rowcount)  
    node  
    [  
        rectangle split,  
        minimum width=1.4cm,  
        rectangle split empty part height=1cm,  
        rectangle split parts=2,  
        rectangle split part fill={cyan!20,magenta!4},  
        draw]  
    {\%d-  
        \nodepart[two]{  
    };  
    \ifdate{Sunday}{\advance\rowcount by 1}{  
    }  
\end{tikzpicture}%  
}
```

↓ Input

7.5 Displaying a Calendar

This produces the calendar shown in [Figure 7.5](#).

Suppose now I want to add some information to the calendar. For example, the two May bank holidays on the 5th and 26th of May. Additionally, suppose I also want a different colour background for weekends and bank holidays, for example, a light grey. The bank holiday information can be stored in control sequences whose names are in the format $\langle\text{prefix}\rangle\text{-}\langle\text{YYYY}\rangle\text{-}\langle\text{MM}\rangle\text{-}\langle\text{DD}\rangle$, which is the format used by `\pgfcalendarsuggestedname`. Recall etoolbox's `\csdef` command described in [§2.1.1](#). This can be used to define these control sequences:

↑ Input

```
\csdef{cal-2014-05-05}{Early May BH}  
\csdef{cal-2014-05-26}{Spring BH}
```

↓ Input

The `\pgfcalendar` command now needs `cal` as the $\langle\text{prefix}\rangle$ so that in the $\langle\text{code}\rangle$ part, the current day can be checked if it's a bank holiday using:

↑ Input

```
\ifcsdef{\pgfcalendarsuggestedname}{%  
% Current day is a~bank holiday
```

7.5 Displaying a Calendar

Output

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Figure 7.5 Calendar with Split Nodes (May 2014)

7.5 Displaying a Calendar

```
}%  
{  
    % Current day isn't a~bank holiday  
}  
|
```

↓ Input

As before a TeX register called `\rowcount` is defined using:

```
\newcount\rowcount
```

Input

The actual code to generate the calendar is now:

```
\fbox{  
    \rowcount=1\relax  
    \let\%{\pgfcalendarshorthand}  
    \begin{tikzpicture}[x=1.5cm,y=1.75cm]  
        \begin{scope}  
            [every node/.style={rectangle,fill=green!5,minimum width=1.4cm}]  
            \path (0,0) node {Mon};  
            \path (1,0) node {Tue};  
            \path (2,0) node {Wed};  
            \path (3,0) node {Thu};
```

↑ Input

7.5 Displaying a Calendar

```
\path (4,0) node {Fri};  
\path (5,0) node {Sat};  
\path (6,0) node {Sun};  
\end{scope}  
\pgfcalendar{cal}{2014-05-01}{2014-05-31}{%  
  \def\thebackground{magenta!4}%  
  \ifcsdef{\pgfcalendarsuggestedname}{%  
    {%  
      \def\thecontents{\csuse{\pgfcalendarsuggestedname}}%  
      \def\thebackground{black!4}%  
    }%  
    {%  
      \def\thecontents{\mbox{}}%  
      \ifdate{weekend}{\def\thebackground{black!4}}{}%  
    }%  
  }\path (\pgfcalendarcurrentweekday,-\rowcount)  
node  
[  
  rectangle split,  
  rectangle split parts=2,  
  rectangle split part fill={cyan!20,\thebackground},  
  draw]
```

7.5 Displaying a Calendar

```
{\%d-
  \nodepart{two}%
  \parbox[t][1cm]{1.2cm}{\small\thecontents}%
} ;
\ifdate{Sunday}{\advance\rowcount by 1}{}
\end{tikzpicture}
}
```

↓ Input

This produces the calendar shown in Figure 7.6.

Suppose now you want to fill in the gaps at the beginning and end of the month. Recall the `\foreach` command mentioned in §2.7.2. This has the syntax:

`\foreach <variables> [<options>] in {<list>} {<body>}`

Definition

but it's cleverer than the other list macros described in that section as you can use ... within `<list>` if the list contents can be inferred from the beginning and end of the list. For example:

`\foreach \x in {1,...,10} {\x\space}`

Input

produces

7.5 Displaying a Calendar

Output

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
Early May BH						
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
Spring BH						

Figure 7.6 May 2014 with Bank Holidays

7.5 Displaying a Calendar

1 2 3 4 5 6 7 8 9 10

Output

Therefore, within the `<code>` part of `\pgfcalendar`, you can test if the current day is the first day of the month (by testing that `\pgfcalendarcurrentday` is equal to 1) and use `\foreach` to fill in the last days of the previous month:

↑ Input

```
\ifnum\pgfcalendarcurrentday=1\relax
% Fill in days from previous month if this isn't a Monday
\ifdate{Monday}{}%
% Get last day of previous month
\julianday = \pgfcalendarcurrentjulian\relax
\advance\julianday by -\pgfcalendarcurrentweekday\relax
\foreach \x in {0,...,\numexpr\pgfcalendarcurrentweekday-1}%
{
  \pgfcalendarjuliantodate{\julianday}{\theyear}{\themonth}{\theday}
  \path (\x,-1)
  node [
    rectangle split,
    rectangle split parts=2,
```

7.5 Displaying a Calendar

```
    draw]
{\number\theday
 \nodepart{two}
 \parbox[t][1cm]{1.2cm}{\mbox{}}
};
\global\advance\julianday by 1\relax
}
}
\fi
```

Input

This requires a new register:

```
\newcount\julianday
```

Input

The tikz package automatically loads the pgffor package, so the `\foreach` command will also be available if you use tikz. Note that `\foreach` uses a local scope for each iteration which is why `\global` is required when incrementing the `\julianday` register. The gap at the end of the final week can also be filled with the initial days of the next month, but as with `\foreach`, `\pgfcalendar` scopes each iteration, so the row register `\rowcount` will need to be incremented globally so that it can be used after the loop has completed. It's also useful to store the Julian day number and the week day number for the last day of the month so they can be accessed

7.5 Displaying a Calendar

outside the loop. This saves the need to compute them again. So the last part of `<code>` needs to replace:

```
\ifdate{Sunday}{\advance\rowcount by 1}{}%
```

Input

with

```
\ifdate{Sunday}{\global\advance\rowcount by 1}{}%
\xdef\lastjulianday{\number\pgfcalendarcurrentjulian}
\xdef\lastweekday{\number\pgfcalendarcurrentweekday}
```

↑ Input

↓ Input

Now the remaining days of the last row can be completed outside the `\pgfcalendar` loop:

```
\ifnum\lastweekday < 6\relax
  \julianday = \lastjulianday\relax
  \edef\lastweekday{\number\numexpr\lastweekday+1}
  \foreach \x in {\lastweekday,...,6}
  {
    \global\advance\julianday by 1\relax
  }
```

↑ Input

7.5 Displaying a Calendar

```
\pgfcalendarjuliantodate{\julianday}{\theyear}{\themonth}{\theday}
\path (\x,-\rowcount)
node [
  rectangle split,
  rectangle split parts=2,
  draw]
{\number\theday
 \nodepart{two}
 \parbox[t]{1cm}{\theday}%
};
\fi
```

↓ Input

Note that you can also use `\foreach` to display the week day nodes:

```
\foreach \x in {0,...,6}
{\path (\x,0) node {\pgfcalendarweekdayshortname{\x}};}
```

↑ Input

↓ Input

7.5 Displaying a Calendar

The complete code is:

↑ Input

```
\newcount\rowcount
\newcount\julianday
\fbox{%
\rowcount=1\relax
\let\%\pgfcalendarshorthand
\begin{tikzpicture}[x=1.5cm,y=1.75cm]
\begin{scope}
[every node/.style={rectangle,fill=green!5,minimum width=1.4cm}]
\foreach \x in {0,...,6}
{\path (\x,0) node {\pgfcalendarweekdayshortname{\x}};}
\end{scope}
\pgfcalendar{cal}{2014-05-01}{2014-05-31}
% Is this the first day of the month?
\ifnum\pgfcalendarcurrentday=1\relax
% Fill in days from previous month if this isn't a Monday
\ifdate{Monday}{}%
% Get last day of previous month
\julianday = \pgfcalendarcurrentjulian\relax
\advance\julianday by -\pgfcalendarcurrentweekday\relax
```

7.5 Displaying a Calendar

```
\foreach \x in {0,...,\numexpr\pgfcalendarcurrentweekday-1}
{
  \pgfcalendarjuliantodate{\julianday}{\theyear}{\themonth}{\theday}
  \path (\x,-1)
    node
      [rectangle split,
       rectangle split parts=2,
       draw]
      {\number\theday
       \nodepart{two}
       \parbox[t][1cm]{1.2cm}{\mbox{}}%
      };
  \global\advance\julianday by 1\relax
}
\fi
\def\thebackground{magenta!4}%
\ifcsdef{\pgfcalendarsuggestedname}%
{%
  \def\thecontents{\csuse{\pgfcalendarsuggestedname}}%
  \def\thebackground{black!4}%
}%

```

7.5 Displaying a Calendar

```
{%
  \def\thecontents{\mbox{}}%
  \ifdate{weekend}{\def\thebackground{black!4}}{}%
}%
\path (\pgfcalendarcurrentweekday,-\rowcount)
node
[rectangle split,
 rectangle split parts=2,
 rectangle split part fill={cyan!20,\thebackground},
 draw]
{\%d-
\nodepart{two}%
\parbox[t][1cm][1.2cm]{\small\thecontents}%
};%
\ifdate{Sunday}{\global\advance\rowcount by 1}{}%
\xdef\lastjulianday{\number\pgfcalendarcurrentjulian}%
\xdef\lastweekday{\number\pgfcalendarcurrentweekday}%
}%
\ifnum\lastweekday < 6\relax
\julianday = \lastjulianday\relax
\edef\lastweekday{\number\numexpr\lastweekday+1}
\foreach \x in {\lastweekday,...,6}
```

7.5 Displaying a Calendar

```
{\global\advance\julianday by 1\relax
\pgfcalendarjuliantodate{\julianday}{\theyear}{\themonth}{\theday}
\path (\x,-\rowcount)
node
[rectangle split,
 rectangle split parts=2,
 draw]
{\number\theday
\nodepart{two}
\parbox[t]{1cm}{1.2cm}{\mbox{}}%
};
}
\fi
\end{tikzpicture}%
}
```

↓ Input

The result is shown in Figure 7.7. You can [download](#) or [view](#) a complete document.

It's possible to create a general month calendar macro from the above.

7.5 Displaying a Calendar

Output

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	1	2	3	4
5	6	7	8	9	10	11
Early May BH						
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
Spring BH						

Figure 7.7 May 2014 with Bank Holidays (including end of previous month and beginning of next month)

7.5 Displaying a Calendar

First a command that can be used to set information for a given date. This uses `\appto` so that information can be appended to a date.

```
\newcommand*\addevent[2]{%
  \ifcsdef{cal-\#1}
    {% already defined so append info
     \csappto{cal-\#1}{\newline #2}%
    }%
  {% not defined
   \csdef{cal-\#1}{#2}%
  }
}
```

↑ Input

↓ Input

This has the syntax:

`\addevent{<date>}{<information>}`

Definition

Now the definition for the calendar month macro:

↑ Input

7.5 Displaying a Calendar

```
\newcommand*{\calendarmonth}[2]{%
  \fbox{%
    \rowcount=1\relax
    \let\%=\pgfcalendarshorthand
    \begin{tikzpicture}[x=1.5cm,y=1.75cm]
      % display the month name at the top
      \path (3,1) node {\pgfcalendarmonthname{#2}};
      \begin{scope}
        [every node/.style={rectangle,fill=green!5,minimum width=1.4cm}]
        \foreach \x in {0,...,6}
          {\path (\x,0) node {\pgfcalendarweekdayshortname{\x}};}
      \end{scope}
    \pgfcalendar{cal}{#1-#2-01}{#1-#2-last}
    {%
      % Is this the first day of the month?
      \ifnum\pgfcalendarcurrentday=1\relax
      % Fill in days from previous month if this isn't a Monday
      \ifdate{Monday}{}{%
        %
        % Get last day of previous month
        \julianday = \pgfcalendarcurrentjulian\relax
        \advance\julianday by -\pgfcalendarcurrentweekday\relax
      }
    }
  }
}
```

7.5 Displaying a Calendar

```
\foreach \x in {0,...,\numexpr\pgfcalendarcurrentweekday-1}
{
  \pgfcalendarjuliantodate
    {\julianday}{\theyear}{\themonth}{\theday}
  \path (\x,-1)
    node
    [
      rectangle split,
      rectangle split parts=2,
      draw]
    {\number\theday
     \nodepart{two}
     \parbox[t][1cm][1.2cm]{\mbox{}}
    };
  \global\advance\julianday by 1\relax
}
\fi
\def\thebackground{magenta!4}%
\ifcsdef{\pgfcalendarsuggestedname}%
{%
  \def\thecontents{\csuse{\pgfcalendarsuggestedname}}%
```

7.5 Displaying a Calendar

```
\def\thebackground{black!4}%
}%
{%
 \def\thecontents{\mbox{}}
 \ifdate{weekend}{\def\thebackground{black!4}}{}%
}%
\path (\pgfcalendarcurrentweekday,-\rowcount)
node
[
  rectangle split,
  rectangle split parts=2,
  rectangle split part fill={cyan!20,\thebackground},
  draw]
{\%d-
 \nodepart{two}%
 \parbox[t][1cm][1.2cm]{\small\thecontents}%
};
\ifdate{Sunday}{\global\advance\rowcount by 1}{}%
\xdef\lastjulianday{\number\pgfcalendarcurrentjulian}%
\xdef\lastweekday{\number\pgfcalendarcurrentweekday}%
}%
\ifnum\lastweekday < 6\relax
```

7.5 Displaying a Calendar

```
\julianday = \lastjulianday\relax
\edef\lastweekday{\number\numexpr\lastweekday+1}
\foreach \x in {\lastweekday,...,6}
{
  \global\advance\julianday by 1\relax
  \pgfcalendarjuliantodate{\julianday}{\theyear}{\themonth}{\theday}
  \path (\x,-\rowcount)
    node
    [
      rectangle split,
      rectangle split parts=2,
      draw]
    {\number\theday
     \nodepart{two}
     \parbox[t][1cm]{1.2cm}{\mbox{}}
    };
}
\fi
\end{tikzpicture}%
}
```

↓ Input

7.5 Displaying a Calendar

The syntax for this macro is:

```
\calendarmonth{\langle YYYY\rangle}{\langle MM\rangle}
```

Definition

Don't forget you also need to define the registers:

```
\newcount\rowcount  
\newcount\julianday
```

↑ Input

↓ Input

EXERCISE 23. CALENDAR FOR 2014

Create a landscape document that has a calendar month per page for 2014 (or the year of your choice). Read the tikz chapter of the pgf manual [102] to find ways of modifying the above code. You can [download](#) or [view](#) a solution.



[FAQ: Producing presentations (including slides)]

8. PRESENTATIONS (THE beamer CLASS)

There are a number of classes listed on the presentation topic page for presentations. For brevity, this book will only cover the beamer class [103]. At the time of writing, the beamer manual is over 200 pages long. This book is already larger than the previous two volumes combined, so this chapter will only look at how to create a basic document using beamer to help you get started.

A document that uses the beamer class typically consists of a series of frame environments. Each frame produces a slide, or possibly several slides if there are overlays. Here's a simple document that just creates a title slide:

```
\documentclass{beamer}

\title{Culinary Experimental Research}
\author{Mabel Canary}
```

↑ Input

Chapter 8. Presentations (The beamer Class)

```
\date{22nd March 2014}
```

```
\begin{document}
  \begin{frame}
    \maketitle
  \end{frame}
\end{document}
```

↓ Input

The beamer class provides an optional argument to `\title`, `\author` and `\date` that isn't available with standard classes, such as article. The optional argument can be used to supply an abbreviated version which may be used in headlines or footlines. There are additional title page commands:

```
\subtitle[<short title>]{<title>}
```

Definition

This specifies a subtitle.

```
\titlegraphic{<graphic>}
```

Definition

This specifies graphics for the title page. Typically, `<graphic>` is code to load an image file.

```
\institute[<short name>]{<name>}
```

Definition

This specifies the author's affiliation. If there are multiple authors from different institutes, the institutes should be separated by `\and` (in a similar manner to `\author`). Additionally, when there are multiple institutes, each institute should be prefixed by

`\inst{<text>}`

Definition

This command should also be placed after the corresponding name (or names) in the argument of `\author` with matching `<text>`.

EXAMPLE:

↑ Input

```
\documentclass{beamer}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\title{Culinary Experimental Research}
\subtitle{Mind-Controlling Cookies and Exploding Chocolates}
\author[Canary and Zebra]{Mabel Canary\inst{1} \and
Zöe Zebra\inst{2}}
\date[Mar'14]{22nd March 2014}
```

```
\institute{\inst{1}Secret Lab of Experimental Stuff\\
  University of Somewhere\and
  \inst{2}Department of Stripy Confectioners\\
  College of Somewhere Else
}
\titlegraphic{\includegraphics[width=1in]{dummy-logo}}
\begin{document}
  \begin{frame}
    \maketitle
  \end{frame}
\end{document}
```

↓ Input

This uses the sample `dummy-logo.png` file available from the examples page. The resulting slide is shown in Figure 8.1. The footline (bottom right of the slide) provides navigation links.

Within the `frame` environment you can use:

```
\frametitle{(title)}
```

Definition

to specify the title of the frame and

Culinary Experimental Research
Mind-Controlling Cookies and Exploding Chocolates

Mabel Canary¹ Zöe Zebra²

¹Secret Lab of Experimental Stuff
University of Somewhere

²Department of Stripy Confectioners
College of Somewhere Else

22nd March 2014

DUMMY
LOGO



Figure 8.1 Title Frame

`\framesubtitle{<subtitle>}`

Definition

to specify the subtitle.

EXAMPLE:

```
\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Ingredients}

  \begin{itemize}
  \item Self-raising flour;
  \item Butter;
  \item Chocolate chips;
  \item Sugar obtained from secret genetically modified beet.
  \end{itemize}
\end{frame}
```

↑ Input

↓ Input

The resulting frame is shown in Figure 8.2.

⚠ The contents of the `frame` environment are fragile. If you want verbatim text in a frame (for example, using `\verb` or the `verbatim` or `lstlisting`

Mind-Controlling Cookies

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;
- ▶ Chocolate chips;
- ▶ Sugar obtained from secret genetically modified beet.



Figure 8.2 An Example Frame

environments) you must use the `fragile` option to the `frame` environment. For example:

```
\begin{frame}[fragile]
\frametitle{Hello World!}
\begin{verbatim}
#!/usr/bin/perl
print "Hello World!\n";
1;
\end{verbatim}
\end{frame}
```

↑ Input

↓ Input

The resulting frame is shown in [Figure 8.3](#).

The standard sectioning commands may be used outside the `frame` environment. These will be added to the PDF bookmarks and you can add a table of contents frame by putting `\tableofcontents` inside a `frame` environment.

EXAMPLE:



Figure 8.3 Verbatim in a Frame

↑ Input

```
\begin{frame}
  \tableofcontents
\end{frame}

\section{Experimental Research}
\subsection{Cookies}

\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Ingredients}

  \begin{itemize}
  \item Self-raising flour;
  \item Butter;
  \item Chocolate chips;
  \item Sugar obtained from secret genetically modified beet.
  \end{itemize}
\end{frame}
```

↓ Input

Areas of a frame can be divided into titled blocks using the `block` environment.

`\begin{block}{<title>}`

Definition

EXAMPLE:

```
\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Recipe}

  \begin{block}{Ingredients}
    \begin{itemize}
      \item Self-raising flour;
      \item Butter;
      \item Chocolate chips;
      \item Sugar obtained from secret genetically modified beet.
    \end{itemize}
  \end{block}
\end{frame}
```

↑ Input

↓ Input

8.1 Overlays

The resulting frame is shown in [Figure 8.4](#).

There are some predefined block environments, such as `theorem` and `proof`.

`\begin{theorem}[\langle title\rangle]`

Definition

If `\langle title\rangle` is present, this is appended to the “Theorem” block title.

`\begin{proof}[\langle proof name\rangle]`

Definition

If `\langle proof name\rangle` is present, it’s used instead of “Proof” as the block title.

8.1 ■ Overlays

Overlays allow you to uncover parts of a slide. For example, to uncover the items in the above `itemize` environment:

```
\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Recipe}
```

↑ Input

8.1 Overlays

Mind-Controlling Cookies

Recipe

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;
- ▶ Chocolate chips;
- ▶ Sugar obtained from secret genetically modified beet.



Figure 8.4 Block

8.1 Overlays

```
\begin{block}{Ingredients}
  \begin{itemize}[<-->]
    \item Self-raising flour;
    \item Butter;
    \item Chocolate chips;
    \item Sugar obtained from secret genetically modified beet.
  \end{itemize}
\end{block}
\end{frame}
```

↓ Input

This creates five slides. The first just has one item (Figure 8.5), the second has two items (Figure 8.6), etc.

This completely hides the text until it's uncovered. If you prefer to show the text faintly before it's uncovered, you can use:

```
\setbeamercovered{transparent}
```

Input

This mixes 85% of the background colour with 15% of the text colour. Note that the effect varies according to the display device. Now the first slide of:

↑ Input

8.1 Overlays

Mind-Controlling Cookies

Recipe

Ingredients

- ▶ Self-raising flour;



Figure 8.5 Overlays (First Slide)

8.1 Overlays

Mind-Controlling Cookies

Recipe

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;

The slide features a navigation bar at the bottom with icons for back, forward, search, and other presentation controls.

Figure 8.6 Overlays (Second Slide)

8.1 Overlays

```
\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Recipe}

  \begin{block}{Ingredients}
    \begin{itemize}[<+->]
      \item Self-raising flour;
      \item Butter;
      \item Chocolate chips;
      \item Sugar obtained from secret genetically modified beet.
    \end{itemize}
  \end{block}
\end{frame}
```

↓ Input

faintly shows the second item onwards (see [Figure 8.7](#)).

Alternatively, you can specify the overlay information for each item.

EXAMPLE:

```
\begin{frame}
  \frametitle{Mind-Controlling Cookies}
```

↑ Input

8.1 Overlays

Mind-Controlling Cookies

Recipe

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;
- ▶ Chocolate chips;
- ▶ Sugar obtained from secret genetically modified beet.



Figure 8.7 Overlays (transparent option)

8.1 Overlays

```
\framesubtitle{Recipe}

\begin{block}{Ingredients}
\begin{itemize}
\item<1-> Self-raising flour;
\item<2-> Butter;
\item<1-> Chocolate chips;
\item<2-> Sugar obtained from secret genetically
modified beet.
\end{itemize}
\end{block}
\end{frame}
```

↓ Input

This shows the first and third items on the first slide and all items on the second slide. You can similarly apply overlays to beamer environments, such as `block`, or to standard environments, such as `enumerate`:

```
\begin{frame}
\frametitle{Mind-Controlling Cookies}
\framesubtitle{Recipe}
```

↑ Input

8.1 Overlays

```
\begin{block}{Ingredients}
  \begin{enumerate}[<+->]
    \item Self-raising flour;
    \item Butter;
    \item Chocolate chips;
    \item Sugar obtained from secret genetically modified beet.
  \end{enumerate}
\end{block}
\end{frame}
```

↓ Input

The general syntax of the overlay specification is $\langle start \rangle - \langle end \rangle$, where $\langle start \rangle$ is the starting index and $\langle end \rangle$ is the end index. Parts of the specification can be omitted. For example, $<2->$ means slide 2 onwards whereas $<2>$ means only on slide 2. This specification can also be applied to some common commands, such as

`\includegraphics<(overlay)>[<options>]{<image file>}`

Definition

For full syntax, see the beamer manual [103].

8.2 Themes

8.2 Themes

The appearance of the slides is governed by *themes*. There are five types of theme:

1. Presentation

A presentation theme governs the whole appearance of the presentation. A presentation theme is chosen with:

`\usetheme[<options>]{<name>}`

Definition

2. Color

A color theme governs the presentation's colour scheme. A color theme is chosen with:

`\usecolortheme[<options>]{<name>}`

Definition

3. Font

A font theme governs the presentation's fonts or font attributes. A font theme is chosen with:

8.2 Themes

`\usefonttheme[options]{name}`

Definition

4. Inner

An inner theme governs the appearance of elements that are considered “inside” a frame. (For example, the appearance of theorems or list items.) An inner theme is chosen with:

`\useinnertheme[options]{name}`

Definition

5. Outer

An outer theme governs the appearance of elements outside a frame. (For example, the headlines or footlines or whether there is a sidebar.) An outer theme is chosen with:

`\useoutertheme[options]{name}`

Definition

The examples above used the default presentation theme. There are a large number of themes to choose from. The rest of this section shows a selection of these themes. The selection is partly influenced by my own preferences, but partly by how well they appear in grey scale for the printed version of this book.

8.2 Themes

EXAMPLE 43. PRESENTATION THEMES (BOADILLA)

```
\documentclass{beamer}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\usetheme{Boadilla}

\titlerunning{Culinary Experimental Research}
\title{Culinary Experimental Research}
\subtitle{Mind-Controlling Cookies and Exploding Chocolates}
\author{Canary and Zebra}{Mabel Canary\inst{1} \and
Zöe Zebra\inst{2}}
\date[Mar'14]{22nd March 2014}
\institute[SLES \& DSC]{\inst{1}Secret Lab of Experimental Stuff\\
University of Somewhere\and
\inst{2}Department of Stripy Confectioners\\
College of Somewhere Else
}
\titlerunning{Culinary Experimental Research}
\titlegraphic{\includegraphics[width=1in]{dummy-logo}}
```

↑ Input

8.2 Themes

```
\begin{document}
  \begin{frame}
    \maketitle
  \end{frame}
\end{document}

\section{Experimental Research}
\subsection{Cookies}

\begin{frame}
  \frametitle{Mind-Controlling Cookies}
  \framesubtitle{Recipe}

  \begin{block}{Ingredients}
  \begin{itemize}
    \item Self-raising flour;
    \item Butter;
    \item Chocolate chips;
    \item Sugar obtained from secret genetically modified beet.
  \end{itemize}
  \end{block}
\end{frame}
```

8.2 Themes

```
\end{frame}
```

↓ Input

This document uses the Boadilla presentation theme. The resulting slides are shown in Figures 8.8 and 8.9. This theme puts the author names, affiliations, title and date in the footline. You can [download](#) or [view](#) this document.

In the above example, if

```
\usepackage{Boadilla}
```

Input

is changed to

```
\usepackage{EastLansing}
```

Input

then the resulting slides have a green colour scheme with a headline (shown in Figures 8.10 and 8.11).

The Montpellier theme shows the sectioning information as a tree in the headline (see Figures 8.12 and 8.13) or there's the Goettingen theme that has a sidebar with a table of contents (see Figures 8.14 and 8.15).

You can mix a presentation theme with the other types of themes. For example, if I replace the line

8.2 Themes

The title slide features a white background with a thin black border. At the top center, the text "Culinary Experimental Research" is displayed in a large, bold, blue font. Below it, the subtitle "Mind-Controlling Cookies and Exploding Chocolates" is shown in a smaller, regular blue font. In the center, the names "Mabel Canary¹" and "Zöe Zebra²" are listed side-by-side. Below each name is a corresponding affiliation: "¹Secret Lab of Experimental Stuff" and "University of Somewhere" under Mabel, and "²Department of Stripy Confectioners" and "College of Somewhere Else" under Zöe. The date "22nd March 2014" is centered below the affiliations. A large, stylized "DUMMY LOGO" is positioned in the lower-left quadrant. The bottom of the slide has a dark blue footer bar containing the text "Canary and Zebra (SLES & DSC)", "Culinary Experimental Research", "Mar '14", and "1 / 2". To the right of the footer bar are standard Beamer navigation icons.

Figure 8.8 Boadilla Theme (Title Slide)

8.2 Themes

Mind-Controlling Cookies

Recipe

Ingredients

- Self-raising flour;
- Butter;
- Chocolate chips;
- Sugar obtained from secret genetically modified beet.



Canary and Zebra (SLES & DSC) Culinary Experimental Research Mar '14 2 / 2

Figure 8.9 Boadilla Theme

8.2 Themes

The title slide features a dark green header bar at the top. Below it is a light green main area containing the title and subtitle. The footer includes navigation icons and a date.

Culinary Experimental Research
Mind-Controlling Cookies and Exploding Chocolates

Mabel Canary¹ Zöe Zebra²

¹Secret Lab of Experimental Stuff
University of Somewhere

²Department of Stripy Confectioners
College of Somewhere Else

22nd March 2014

DUMMY
LOGO

Canary and Zebra (SLES & DSC) Culinary Experimental Research Mar'14 1 / 2

Figure 8.10 EastLansing Theme (Title Slide)

8.2 Themes

The screenshot shows a presentation slide with the following details:

- Theme:** Experimental Research / Cookies
- Title:** Mind-Controlling Cookies
- Section:** Recipe
- Content:** Ingredients
 - Self-raising flour;
 - Butter;
 - Chocolate chips;
 - Sugar obtained from secret genetically modified beet.
- Navigation:** Includes standard presentation navigation icons (back, forward, search).
- Page Information:** Canary and Zebra (SLES & DSC), Culinary Experimental Research, Mar'14, 2 / 2.

Figure 8.11 EastLansing Theme

8.2 Themes

Culinary Experimental Research

Culinary Experimental Research

Mind-Controlling Cookies and Exploding Chocolates

Mabel Canary¹ Zöe Zebra²

¹Secret Lab of Experimental Stuff
University of Somewhere

²Department of Stripy Confectioners
College of Somewhere Else

22nd March 2014

DUMMY
LOGO

Figure 8.12 Montpellier Theme (Title Slide)

8.2 Themes

Culinary Experimental Research
└ Experimental Research
 └ Cookies

Mind-Controlling Cookies

Recipe

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;
- ▶ Chocolate chips;
- ▶ Sugar obtained from secret genetically modified beet.

Navigation icons: back, forward, search, etc.

Figure 8.13 Montpellier Theme

8.2 Themes

The title slide features a white main area and a light blue sidebar on the right. The sidebar contains the text 'Culinary Experimental Research', 'Canary and Zebra', 'Experimental Research', and 'Cookies'. The main content includes the title 'Culinary Experimental Research', subtitle 'Mind-Controlling Cookies and Exploding Chocolates', author names 'Mabel Canary¹ Zöe Zebra²', and dates '22nd March 2014'. A 'DUMMY LOGO' placeholder is at the bottom. Navigation icons are at the bottom of the slide.

Culinary
Experimental
Research

Canary and
Zebra

Experimental
Research

Cookies

Culinary Experimental Research

Mind-Controlling Cookies and Exploding Chocolates

Mabel Canary¹ Zöe Zebra²

¹Secret Lab of Experimental Stuff
University of Somewhere

²Department of Stripy Confectioners
College of Somewhere Else

22nd March 2014

DUMMY
LOGO

Figure 8.14 Goettingen Theme (Title Slide)

8.2 Themes

Mind-Controlling Cookies

Recipe

Culinary Experimental Research

Canary and Zebra

Experimental Research

Cookies

Ingredients

- ▶ Self-raising flour;
- ▶ Butter;
- ▶ Chocolate chips;
- ▶ Sugar obtained from secret genetically modified beet.

Navigation icons: back, forward, search, etc.

Figure 8.15 Goettingen Theme

8.2 Themes

\usetheme{Boadilla}

Input

from Example 43 with

```
\usetheme{Goettingen}
\useinnertheme[shadow]{rounded}
\usecolortheme{spruce}
```

↑ Input

↓ Input

then the result is as shown in Figures 8.16 and 8.17.

8.2 Themes

The title slide features a light green header bar with the text 'Culinary Experimental Research' and a subtitle 'Mind-Controlling Cookies and Exploding Chocolates'. Below the header, the names 'Mabel Canary¹' and 'Zoe Zebra²' are listed. A note below 'Canary' specifies 'Secret Lab of Experimental Stuff' at 'University of Somewhere'. A note below 'Zebra' specifies 'Department of Stripy Confectioners' at 'College of Somewhere Else'. The date '22nd March 2014' is centered below the names. At the bottom is a placeholder 'DUMMY LOGO' with a blue circular icon. The right side of the slide has a vertical purple sidebar containing the text 'Culinary Experimental Research', 'Canary and Zebra', and 'Experimental Research Cookies'. The bottom of the slide includes standard presentation navigation icons.

Culinary Experimental Research
Mind-Controlling Cookies and Exploding Chocolates

Mabel Canary¹ Zoe Zebra²

¹Secret Lab of Experimental Stuff
University of Somewhere

²Department of Stripy Confectioners
College of Somewhere Else

22nd March 2014

DUMMY
LOGO

Culinary
Experimental
Research

Canary and
Zebra

Experimental
Research
Cookies

Figure 8.16 Goettingen Presentation Theme with rounded Inner Theme and spruce Color Theme (Title Slide)

8.2 Themes

Mind-Controlling Cookies
Recipe

Culinary Experimental Research
Canary and Zebra

Experimental Research
Cookies

Ingredients

- Self-raising flour;
- Butter;
- Chocolate chips;
- Sugar obtained from secret genetically modified beet.

Navigation icons at the bottom include: back, forward, search, and other presentation controls.

Figure 8.17 Goettingen Presentation Theme with rounded Inner Theme and spruce Color Theme

9. ■ ASSIGNMENTS AND EXAMINATIONS

There are a number of classes or packages available on [CTAN](#) to help typeset assignment sheets or exam papers (see the [exam topic](#)).

The exam class (version 2.4, 2011-05-22) is comprehensive with its own list-like environments for enumerating questions and their parts, as well as environments for multiple choice questions. It also provides the possibility of assigning points to each question (or question part), creating solution sheets and grading tables. The exam class is described in §9.1, but take care as there is also an exam class provided by the exams bundle. The exam class described in this book is the one by Philip S. Hirschhorn and is in both the TeX Live and MiK^TE_X distributions. The exams bundle isn't in either of those distributions.

The exsheets package (version 0.17, 2014-10-15) also provides a means to create questions and their solutions. Unlike the exam class, the exsheets package allows you to divide the questions into classes and they can be printed selectively. Meta-data can also be assigned to the questions. The solutions may be either printed with their question or collected and printed together at a later point in the document. However, the package doesn't

support multiple choice style of questions (although you can use a package such as `paralist` to create inline numbered lists). The `exsheets` package is described in §9.2.

The `probsoln` package (version 3.04, 2012-08-23) provides a way to define problems and their solutions in a file (or multiple files). You can load all problems, a specific list of problems or a random selection. You may have one or more problem datasets and either print the solutions with the questions or print the solutions later on in the document. The `probsoln` package provides an inline numbered list environment. The `probsoln` package is described in §9.3.

The `datatool` package used in earlier chapters can also be used to store questions and their solutions. The `datatooltk` application can import the dataset files used by `probsoln`. You can add extra fields to specify, for example, the difficulty level or subject. §9.4 describes how you can use `datatool` to create a database of exam or assignment questions and their solutions.

Note that it's possible to use the `exam` class with either the `probsoln` package or the `datatool` package, but it can't be used with the `exsheets` package.

 This chapter assumes that all your files are stored in a secure location that can't be accessed by curious students. If your operating system supports owner-only file permissions, I suggest you set them as appropriate (for example `chmod 600`), but don't rely on that as your only security measure (see also §2.3).

9.1 The exam Class

The exam class is a comprehensive class for typesetting examination papers. The user guide [36] is over 100 pages, so this section is only intended as a brief introduction. The exam class has the standard class options, such as 10pt, 11pt and 12pt, but also provides two other options `addpoints` and `answers`, described below.

The `addpoints` class option enables the point-totalling commands. If this option is used, all points must be specified as integers. Half points may be indicated with the command:

`\half`

Definition

For example, `2\half` instead of 2.5.

In addition to the `addpoints` class option, you can also switch this mode on and off using the commands:

`\addpoints`

Definition

(to switch it on) and

`\noaddpoints`

Definition

(to switch it off).

9.1 The *exam* Class

The *answers* class option will display solutions. Again there are commands to switch this mode on and off:

`\printanswers`

Definition

(to show the solutions) and

`\noprintanswers`

Definition

(to hide the solutions).

All the exam questions should be contained within the single *questions* environment.

`\begin{questions}`
(exam questions)
`\end{questions}`

Definition

Within the *questions* environment, you start a new question using:

`\question[(points)]`

Definition

where *(points)* is the number of points this question is worth. As mentioned above, if the *addpoints* option is on, *(points)* must an integer with optionally the `\half` command (or just the `\half` command on its own). The question is automatically numbered.

You can replace `\question[(points)]` with

9.1 The exam Class

\titledquestion{<title>}[<points>]

Definition

if question should have a title. (The question number won't be included by default, but you can add it to the title using \thequestion in the <title> argument.)

The exam class user guide [36] suggests the following code if you want to provide space for the student to fill in their name if they are to write on the question sheet:

↑ Input

```
\begin{center}
\fbox{\parbox{5.5in}{\centering
Answer the questions in the spaces provided on the
question sheets. If you run out of room for an answer,
continue on the back of the page.%}}
\end{center}

\vspace{0.1in}

\makebox[\textwidth]{Name and section:\enspace\hrulefill}
```

9.1 The *exam* Class

```
\vspace{0.2in}
```

```
\makebox[\textwidth]{Instructor's name:\enspace\hrulefill}
```

↓ Input

(Although you may prefer to use a fraction of `\textwidth` instead of the hardcoded 5.5in.)

EXAMPLE:

```
\documentclass[addpoints]{exam}
```

↑ Input

```
\begin{document}
\begin{center}
\fbox{\parbox{0.8\textwidth}{\centering
Answer the questions in the spaces provided on the
question sheets. If you run out of room for an answer,
continue on the back of the page.%}}
```

```
}}
\end{center}
```

```
\vspace{0.1in}
```

9.1 The *exam* Class

```
\noindent\makebox[\textwidth]{Name and section:\enspace\hrulefill}  
  
\vspace{0.2in}  
  
\noindent\makebox[\textwidth]{Instructor's name:\enspace\hrulefill}  
\begin{questions}  
  \question[3\half] What ingredients can be found in  
    mind-controlling cookies?  
  
  \question[2\half] What are the health benefits of  
    exploding chocolates?  
\end{questions}  
\end{document}
```

↓ Input

This produces:

↑ Output

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

9.1 The *exam* Class

Name and section: _____

Instructor's name: _____

1. (3½ points) What ingredients can be found in mind-controlling cookies?
2. (2½ points) What are the health benefits of exploding chocolates?

↓ Output

Questions may have parts, sub-parts or sub-sub-parts. Question parts should be enclosed in the `parts` environment:

```
\begin{parts}  
<question parts>  
\end{parts}
```

Definition

where each part is started with

```
\part[<points>]
```

Definition

Note that outside of the `parts` environment this command behaves as the standard `\part` sectioning command.

Sub-parts should be enclosed in the `subparts` environment:

9.1 The `exam` Class

```
\begin{subparts}  
<question sub-parts>  
\end{subparts}
```

Definition

where each sub-part is started with

```
\subpart[<points>]
```

Definition

Sub-sub-parts should be enclosed in the `subsubparts` environment:

```
\begin{subsubparts}  
<question sub-sub-parts>  
\end{subsubparts}
```

Definition

where each sub-sub-part is started with

```
\subsubpart[<points>]
```

Definition

As with `\question`, the optional argument indicates the number of points the part is worth, and should only contain digits or `\half` if the `addpoints` option is on.

EXAMPLE:

```
\begin{questions}
```

↑ Input

9.1 The exam Class

\question Find the derivatives with respect to x of the following functions:

```
\begin{parts}
\part[\frac{1}{2}] $y = x + 1$ 
\part[1] $y = x^3 + 4x^2 - x + 3$ 
\part[1\frac{1}{2}] $y = \cos(x^2)$ 
\end{parts}
\end{questions}
```

↓ Input

The result is:

↑ Output

1. Find the derivatives with respect to x of the following functions:

- (a) ($\frac{1}{2}$ point) $y = x + 1$
- (b) (1 point) $y = x^3 + 4x^2 - x + 3$
- (c) ($1\frac{1}{2}$ points) $y = \cos(x^2)$

↓ Output

If you want to specify questions worth bonus points you can use

9.1 The *exam* Class

\bonusquestion

Definition

instead of \question,

\bonustitledquestion

Definition

instead of \titledquestion,

\bonuspart

Definition

instead of \part,

\bonussubpart

Definition

instead of \subpart and

\bonussubsubpart

Definition

instead of \subsubpart.

The default location of the points is after the question (or part) number.
This can be changed to the margin using the declaration

\pointsinmargin

Definition

which puts the points in the left margin, or the declaration

9.1 The *exam* Class

\pointsinrightmargin

Definition

which puts the points in the right margin. If you prefer to place the points elsewhere, use the declaration

\pointsdroppedatright

Definition

to switch off the automatic placement of the points and use

\droppoints

Definition

at the end of the paragraph where you want the points displayed. If you use \pointsdroppedatright but don't use \droppoints the points won't be displayed. Note that \droppoints should only occur at the end of a paragraph or between paragraphs.

The number of questions, parts, sub-parts and sub-sub-parts can be referenced with the commands:

\numquestions

Definition

(the number of questions),

\numparts

Definition

(the number of parts),

9.1 The *exam* Class

`\numsubparts`

Definition

(the number of sub-parts), and

`\numsubsubparts`

Definition

(the number of sub-sub-parts).

If you have used the `addpoints` option, you can reference the total number of points using:

`\numpoints`

Definition

For example:

```
This exam has \numquestions\_\_questions worth a total  
of \numpoints\_\_points.
```

↑ Input

↓ Input

With the `addpoints` option, you can also display a grading table using:

`\gradetable[⟨orientation⟩][⟨index type⟩]`

Definition

where `⟨orientation⟩` may be either `h` (horizontal) or `v` (vertical) and `⟨index type⟩` may be either `questions` or `pages`. The defaults are a vertical table

9.1 The *exam* Class

indexed by question. The grading table provides space for the student marks to be written in by hand. You will need at least two L^AT_EX runs (possibly three or four) to ensure the grading table is up-to-date.

If you want to centre the grading table, you can place it in the `center` environment, which will add extra vertical space at the start and end as well as centre its contents. (It's unlikely that you'll want the grading table in a floating environment as it typically appears in a fixed location.) The grading table is placed in a `tabular` environment, so it can't break across a page.

Any cover page material can be placed in the `coverpages` environment.

```
\begin{coverpages}  
<text>  
\end{coverpages}
```

Definition

The cover pages use Roman numeral page numbers. The page counter is reset at the end of the environment. This environment must not contain the `questions` environment.

There are four environments for typesetting the solutions. The default behaviour is for the solutions to be hidden. To display the solutions, use the `answers` class option. Each environment takes an optional argument, which is the amount of space to be left for the students to write in their solution. If the optional argument is omitted, all four environments simply ignore their contents when the `answers` option hasn't been set.

9.1 The *exam* Class

```
\begin{solution}[\langle length \rangle]  
<solution text>  
\end{solution}
```

Definition

If the solutions are hidden, this environment inserts $\langle length \rangle$ amount of blank space (as though you had used `\vspace*{\langle length \rangle}`).

```
\begin{solutionorbox}[\langle length \rangle]  
<solution text>  
\end{solutionorbox}
```

Definition

The `solutionorbox` environment is similar to the `solution` environment, but inserts an empty box of height $\langle length \rangle$ if the answers should be hidden.

```
\begin{solutionorlines}[\langle length \rangle]  
<solution text>  
\end{solutionorlines}
```

Definition

The `solutionorlines` environment is similar to the `solution` environment, but inserts an area of height $\langle length \rangle$ with ruled lines if the answers should be hidden.

```
\begin{solutionordottedlines}[\langle length \rangle]  
<solution text>  
\end{solutionordottedlines}
```

Definition

9.1 The *exam* Class

The `solutionordottedlines` environment is similar to `solutionorlines` but uses dotted lines.

EXAMPLE:

↑ Input

```
\begin{questions}
\question Find the derivatives with respect to \$x\$ of the
following functions:
\begin{parts}
\part[\half] $y = x + 1$ 

\begin{solution}
$y' = 1$ 
\end{solution}

\part[1] $y = x^3 + 4x^2 - x + 3$ 

\begin{solution}
$y' = 3x^2 + 8x - 1$ 
\end{solution}
```

9.1 The exam Class

```
\part[1\half] $y = \cos(x^2)$  
  
\begin{solution}  
$y' = -2x\sin(x^2)$  
\end{solution}  
  
\end{parts}  
\end{questions}
```

↓ Input

There are four environments provided for multiple choice questions. The first two label the choices and the other two print checkboxes in front of the choices for the student to tick. With these environments use:

`\choice`

Definition

to start a new choice. To indicate the correct choice, use

`\CorrectChoice`

Definition

instead of `\choice`.

```
\begin{choices}  
(list items)  
\end{choices}
```

Definition

9.1 The *exam* Class

The `choices` environment is a list environment with labelled choices as the items in the list.

```
\begin{oneparchoices}  
<list items>  
\end{oneparchoices}
```

Definition

The `oneparchoices` is an in-line list of labelled choices where there are no paragraph breaks unless explicitly inserted.

```
\begin{checkboxes}  
<list items>  
\end{checkboxes}
```

Definition

The `checkboxes` environment is a list environment with checkbox choices as the items in the list.

```
\begin{oneparcheckboxes}  
<list items>  
\end{oneparcheckboxes}
```

Definition

The `oneparcheckboxes` is an in-line list of checkbox choices where there are no paragraph breaks unless explicitly inserted.

EXAMPLE:

9.1 The exam Class

↑ Input

```
\begin{questions}
\question[1] Which of the following ingredients are used in
mind-controlling cookies:
\begin{choices}
\choice arsenic
\choice cyanide
\choice curare
\CorrectChoice secret genetically modified sugar beet
\end{choices}
\end{questions}
```

↓ Input

This produces:

↑ Output

1. (1 point) Which of the following ingredients are used in mind-controlling cookies:
 - A. arsenic
 - B. cyanide

9.1 The *exam* Class

- C. curare
- D. secret genetically modified sugar beet

↓ Output

For more details about the *exam* class, including commands and options not mentioned here, see the *exam* class user guide [36].

EXERCISE 24. CREATING AN EXAM PAPER WITH THE *exam* CLASS

Create an examination paper containing the questions (and solutions) in the above examples and add a grading table. Try experimenting with adding and removing the *answers* class option and try the variations of the *solution* environment (such as *solutionorbox*).

The *exam* class user guide [36] describes ways of adjusting the default settings. Try adding the command `\unframedsolutions` to the preamble and see how it changes the way the solutions are displayed or add `\bracketedpoints` to see how it affects the way the points are displayed. You can [download](#) or [view](#) a solution.

9.2 The exsheets Package

Unlike the exam class described in the [previous section](#), exsheets is a package, so you need to find a suitable class to use with it. For the examples, I'm just going to use the article class, but you will probably find it easier to use a more flexible class, such as one of the KOMA-Script classes.

As with the exam class, the exsheets documentation [62] is quite long because there are so many options, so this section is just intended as an introduction. Options can be specified via the package option list, or in the optional argument to environments provided by exsheets, or via

`\SetupExSheets[<module>]{<option list>}`

Definition

where the options listed in `<option list>` belong to the given module. If `<module>` is omitted, the module name is incorporated into the option list. For example, you can either do:

`\SetupExSheets[question]{<option>=<value>}`

Input

or

`\SetupExSheets{question/<option>=<value>}`

Input

where `<option>` is some option defined for the question module and `<value>` is the value being assigned to that option. There are a lot of options related

9.2 The *exsheets* Package

to the formatting of counters, question headings and subtitles, as well as options related to the table of contents.

Questions are written inside the `question` environment.

```
\begin{question}[\langle options \rangle]\{\langle points \rangle  
⟨question text⟩  
\end{question}
```

Definition

Note that the `⟨points⟩` argument is optional, despite the lack of square brackets. The other optional argument, `⟨options⟩`, is the list of options. If `⟨points⟩` is omitted, no points will be associated with the question. The `⟨points⟩` may be in the form `⟨p⟩` or `⟨p⟩+⟨b⟩` or `+⟨b⟩`, where `⟨p⟩` is the number of points and `⟨b⟩` is the number of bonus points.

EXAMPLES:

A question with no points allocated:

```
\begin{question}  
How many kilocalories are there in 100 grammes of exploding  
chocolate?  
\end{question}
```

↑ Input

↓ Input

9.2 The *exsheets* Package

A question worth five points:

```
\begin{question}{5}
How many kilocalories are there in 100 grammes of exploding
chocolate?
\end{question}
```

↑ Input

↓ Input

A question worth five points and one bonus point:

```
\begin{question}{5+1}
How many kilocalories are there in 100 grammes of exploding
chocolate?
\end{question}
```

↑ Input

↓ Input

The points for each question are added to the total marks. If you don't want the points added for a particular question, you need to put an exclamation mark ! before the points. (This prevents bonus points for the question.)

9.2 The *exsheets* Package

EXAMPLE:

A question worth five points where the points aren't added to the running total:

```
\begin{question}{!5}
```

↑ Input

How many kilocalories are there in 100 grammes of exploding
chocolate?

```
\end{question}
```

↓ Input

There are a number of `<key>=<value>` options available for the `question` environment. For a complete list, see the *exsheets* documentation [62], but here are a few:

`type=<value>` Determines the type of question. The value may be either `exam` or `exercise`. In the first case, the question number is preceded by “Question”, in the second case by “Exercise”.

`name=<name>` Replaces the default “Exercise” or “Question” to `<name>`.

`subtitle=<title>` Adds a subtitle.

9.2 The *exsheets* Package

<code>class=<class></code>	Assigns a class to the question.
<code>topic=<topic></code>	Assigns a topic to the question.
<code>ID=<label></code>	Assigns an ID to the question for later reference.
<code>print</code>	A <code>boolean key</code> . If true, the question is displayed. By default this is true.

The solution to a question should come after the `question` environment and should be placed inside the `solution` environment.

```
\begin{solution}[\langle options \rangle]  
⟨solution text⟩  
\end{solution}
```

Definition

As with the `question` environment, the `solution` environment also allows the `print boolean key` in its `\langle options \rangle`, but by default it's false. The `name` key is also available in `\langle options \rangle` and can be used to replace the default "Solution" text that precedes the solution number.

EXAMPLE:

9.2 The *exsheets* Package

```
\begin{question}  
What is the main drawback of ray guns?  
\end{question}  
\begin{solution}  
Overheating.  
\end{solution}
```

↑ Input

↓ Input

By default, the question will appear but the solution won't be displayed.
The default result is:

Exercise 1.

What is the main drawback of ray guns?

↑ Output

↓ Output

If you want the solutions to appear where they are defined (that is, after their associated question), you can just add the following:

```
\SetupExSheets[solution]{print=true}
```

Input

or

9.2 The *exsheets* Package

`\SetupExSheets{solution/print=true}`

Input

However, if you want the solutions to appear later (for example, at the end of the document) you can use

`\printsolutions[⟨settings⟩]`

Definition

at the place where you want the solutions. Note that this command must only be used *after* all the solutions have been defined. The optional argument is a `key=value` list. Options include:

- chapter** If no value is specified, all solutions to the questions defined in the current chapter are listed. If a value is specified, it may be a `comma-separated list` or range of chapter numbers. For example, `chapter={1-7,10}` means chapters one to seven and chapter ten. Remember to use braces around the value if it contains commas.
- section** Analogous to the `chapter` option, but for sections. As above, the value may be omitted, in which case the current section is assumed, or may be a `comma-separated list` or range of values.
- byID** The value should be a `comma-separated list` of IDs identifying the questions whose solutions should be printed. (Recall the `ID` option that can be used when you define a question.)

9.2 The *exsheets* Package

The solutions are sorted automatically according to their order of definition. If you want to prevent this sorting (so that they are, instead, listed in the order specified in the value of `byID`) you can use the `sorted` boolean key in the solution module. The `sorted` key isn't required if the `byID` key isn't used.

Questions can be assigned to a “class”, which could represent the difficulty level or similar attribute. As described earlier, this is done via the `class` key when you define a question. You can specify that only questions belonging to a certain class, or list of classes, should be included using the `use-classes` option. For example:

```
\SetupExSheets{use-classes={easy,medium,hard}}
```

Input

indicates to only use those questions that have been assigned to one of the classes: `easy`, `medium` or `hard`. Any questions that haven't been assigned to one of those classes will be discarded.

Similarly, questions can be assigned to a topic using the `topic` key in the optional argument of the `question` environment. The option `use-topics` is analogous to `use-classes`. There are other commands that allow you to assign properties to questions, but for brevity these are omitted here.

As with the `exam` class, you can also access the total points and, as before, you need at least two L^AT_EX runs to get an up-to-date value.

9.2 The *exsheets* Package

\pointsum

Definition

Prints the total number of points (excluding bonus points). This command also has a starred version that omits the “unit”, which is “P.” by default.

\bonussum

Definition

Prints the total number of bonus points. As with the previous command, this command also has a starred version that omits the unit.

\totalpoints

Definition

Prints the total number of points (including bonus points). This command also has a starred version that omits the unit.

To typeset a number of points without adding it to the cumulative total use:

\points{<number>}

Definition

where `<number>` is the number of points. As above, this command has a starred version that omits the unit.

There are some other related commands described in the *exsheets* manual [62], but for brevity aren’t covered here. The “unit” can be changed via the `name` and `name-plural` keys in the `points` module. Recall from earlier, that the option can be set using:

9.2 The *exsheets* Package

```
\SetupExSheets[points]{name={point},name-plural={points}}
```

Input

or

```
\SetupExSheets{points/name={point},points/name-plural={points}}
```

Input

There are also options for the bonus point unit: `bonus-name` and `bonus-plural`.
For example:

```
\SetupExSheets[points]{bonus-name={bonus point},  
                     bonus-plural={bonus points}}
```

↑ Input

↓ Input

There are some other options related to the formatting of the points. See the *exsheets* manual for further details.

There is no provision for multiple choice questions, however you can use the `inparaenum` environment provided by the `paralist` package [78].

```
\begin{inparaenum}[\langle format \rangle]  
(list items)  
\end{inparaenum}
```

Definition

This is analogous to the `enumerate` environment except that each item doesn't start a new paragraph (unless you explicitly insert a paragraph break).

9.2 The *exsheets* Package

The optional argument determines the counter format. The tokens A, a, I, i and 1 indicate the counter formats \Alph, \alph, \Roman, \roman and \arabic. (The paralist package also modifies the enumerate environment so that it takes an optional argument that changes the counter format in the same manner.)

EXAMPLE:

↑ Input

```
\begin{question}
  Which of the following ingredients are used in
  mind-controlling cookies:
  \begin{inparaenum}[(A)]
    \item arsenic
    \item cyanide
    \item curare
    \item\label{correct-ingredient} secret genetically modified
      sugar beet
  \end{inparaenum}
\end{question}
\begin{solution}
  Correct choice: \ref{correct-ingredient}.
\end{solution}
```

9.2 The *exsheets* Package

```
\end{solution}
```

↓ Input

You can put all your `question` and `solution` environments in an external file to form a databank. These solutions, either all or a subset, can then be included using:

```
\includequestions[<options>]{<filenames>}
```

Definition

where `<filenames>` is a `comma-separated list` of filenames. Note that the *exsheets* documentation comes with a caveat that this command is experimental. The optional argument is a `key=value list`. Available options are:

- all** This is a `boolean key`. If true, all questions are selected.
- IDs** Only those solutions whose ID is contained in the list of IDs. Since this value contains commas, remember to enclose the list with braces.
- random** The value should be a number, `<n>`, indicating that `<n>` questions should be randomly selected.
- exclude** Exclude any questions whose ID is contained in this list. Again, since the value contains commas, remember to enclose the list with braces. This option can be combined with the `random` option.

9.3 The *probsoln* Package

EXERCISE 25. CREATING AN EXAM PAPER WITH THE *exsheets* PACKAGE

Try rewriting [Exercise 24](#) using the *exsheets* package instead of the *exam* class. (You may not be able to automatically implement some features, such as the grading table.) You can [download](#) or [view](#) a solution.

9.3 ■ The *probsoln* Package

The *probsoln* package [94] provides a means to define problems with their associated solution. These definitions may be placed in an external `.tex` file. You can then load all problems or a subset of problems (possibly randomly selected) into a dataset. You can have more than one dataset, each of which could, for example, represent a topic. In your document you can iterate through these datasets and display the problem, the solution or both. This means that you can gather the solutions together in another part of the document. Since *probsoln* is a package, you need to find an appropriate document class.

The *probsoln* package has the following options:

`answers` Show the solutions.

`noanswers` Hide the solutions (default).

9.3 The *probsoln* Package

<code>draft</code>	Display the problem label and dataset name when a problem is used.
<code>final</code>	Don't display the problem label and dataset name (default).
<code>usedefaultargs</code>	Make <code>\thisproblem</code> use the default arguments supplied with the problem definition.
<code>nousedefaultargs</code>	Make <code>\thisproblem</code> prompt for arguments (default).

The last two options will be described in more detail below. Remember that any options specified in `\documentclass` are also passed to packages, so if you use the `draft` class option, it will automatically enable *probsoln*'s `draft` option, unless you have explicitly used *probsoln*'s `final` option.

At the time of writing the current version of *probsoln* is version 3.04 (2012-08-23). Some of the features described here aren't available for earlier versions. Since defining the problems and their solutions requires either a command or an environment that gathers its contents, verbatim code requires special care. To allow for verbatim text, the *probsoln* package provides the `fragile` boolean key that can be used when defining a problem. If the majority of your problems require this option, you can set it using:

9.3 The *probsoln* Package

\setkeys{probsoln}{fragile}

Input

(This command is provided by the keyval package [12], which is automatically loaded.)

In order to work with verbatim code, the *probsoln* package creates a temporary file that's used when the *fragile* option is set. The default name for this file is `\jobname.vrb` but if this conflicts with another package, you can change the extension by redefining

\ProbSolnFragileExt

Definition

You can also change the basename by redefining

\ProbSolnFragileFile

Definition

In addition to the *answers* and *noanswers* options, you can also show or suppress solutions using

\showanswers

Definition

(to show the solutions) and

\hideanswers

Definition

(to hide the solutions). These are declarations that can be scoped by placing them within a group.

9.3 The *probsoln* Package

You can check if the show solutions setting is on or off by testing the `showanswers` boolean flag. This can be done using:

```
\ifshowanswers {true part}\else {false part}\fi
```

Definition

or you can use the `\ifthenelse` command provided by the `ifthen` package:

```
\ifthenelse{\boolean{showanswers}}{<true part>}{<false part>}
```

Definition

or you can use the `\ifbool` command provided by the `etoolbox` package:

```
\ifbool{showanswers}{<true part>}{<false part>}
```

Definition

For example:

```
Assignment 1\ifbool{showanswers}{ (Solution Sheet)}{}
```

Input

For longer text, you can use the environments `onlyproblem`

```
\begin{onlyproblem}[<option>]  
<text>  
\end{onlyproblem}
```

Definition

to only display `<text>` if the solutions are suppressed, and `onlysolution`

```
\begin{onlysolution}[<option>]  
<text>  
\end{onlysolution}
```

Definition

9.3 The *probsoln* Package

to only display $\langle\text{text}\rangle$ if the solutions are displayed. In both cases, the optional argument $\langle\text{option}\rangle$ may be the **fragile boolean key**, described above.

EXAMPLE:

```
\begin{onlyproblem}
What is the main drawback of ray guns?
\end{onlyproblem}
\begin{onlysolution}
Overheating.
\end{onlysolution}
```

↑ Input

↓ Input

If the solutions are displayed, only the solution ("Overheating.") will be typeset, otherwise only the question ("What is the main drawback of ray guns?") will be typeset.

Note that spaces at the start of the environment are discarded but spaces at the end of the environment aren't. So the EOL character immediately before "What" is discarded (and similarly for the EOL character immediately before "Overheating") but the EOL character after the question mark is interpreted as a space (and similarly for the EOL character after the full

9.3 The *probsoln* Package

stop in the solution). If these spaces are unwanted, you can suppress them with the % comment character:

```
\begin{onlyproblem}
What is the main drawback of ray guns?%
\end{onlyproblem}%
\begin{onlysolution}
Overheating.%
\end{onlysolution}
```

↑ Input

↓ Input

If you want the question to always appear, regardless of the `showanswers` flag, then don't put it inside the `onlyproblem` environment:

```
What is the main drawback of ray guns?
\begin{onlysolution}
Overheating.%
\end{onlysolution}
```

↑ Input

↓ Input

9.3 The `probsoln` Package

You may prefer to put the solution inside the `solution` environment:

```
\begin{solution}{text}\end{solution}
```

Definition

which is equivalent to:

```
\par\noindent\textbf{\solutionname}: <text>
```

Input

where `\solutionname` defaults to “Solution”.

EXAMPLE:

```
What is the main drawback of ray guns?
```

↑ Input

```
\begin{onlysolution}
```

```
\begin{solution}
```

Overheating.

```
\end{solution}
```

```
\end{onlysolution}
```

↓ Input

The combination of the `onlysolution` and `solution` environments in this manner is analogous to the `exam` class’s `solution` environment. Since it’s possible that you may want to use the `probsoln` package with the `exam` class, if `probsoln` detects that the `solution` environment is already defined, it doesn’t try

9.3 The *probsoln* Package

defining its own `solution` environment. This means that if the above example was used with both the `exam` class and the `probsoln` package, there's a level of redundancy with the `exam` class performing a check for its own show/hide solution setting in its `solution` environment and the `probsoln` package performing a similar check for its `showanswers` flag in its `onlysolution` environment. Therefore, if you are using both `exam` and `probsoln`, I recommend you redefine the `solution` environment in a manner similar to `probsoln`'s `solution` environment and just use `probsoln`'s `showanswers` flag, as shown in Example 44.

The `probsoln` package provides an inline numbered list environment called `textenum`:

```
\begin{textenum}{items}\end{textenum}
```

Definition

This uses the same counter and label as the `enumerate` environment would use at the current level. For example, if the `textenum` environment was used outside the `enumerate` environment, the `enumi` counter will be used with `\labelenumi`, but if `textenum` was used inside a single `enumerate` environment, the `enumii` will be used with `\labelenumii`.

As with the standard list environments, each item is started with `\item` but no paragraph break is inserted unless you explicitly add one (either via a blank line or `\par`). You can also use

9.3 The *probsoln* Package

`\correctitem`

Definition

in place of `\item` to indicate a correct choice and

`\incorrectitem`

Definition

in place of `\item` to indicate an incorrect choice. If the solutions are suppressed, these two commands behave the same as `\item`. When the solutions are displayed, the default behaviour of `\correctitem` is to put a frame around the item marker, and the default behaviour of `\incorrectitem` is to just display the marker (as per `\item`). You can change this by redefining

`\correctitemformat{\langle marker\rangle}`

Definition

which governs the format used by `\correctitem`, and

`\incorrectitemformat{\langle marker\rangle}`

Definition

which governs the format used by `\incorrectitem`.

EXAMPLE 44. USING BOTH THE `exam` CLASS AND THE *probsoln* PACKAGE

This example uses the `exam` class with the *probsoln* package. The `exam` class's `solution` environment is redefined to make it more like the *probsoln* package's `solution` environment. The `\ignorespaces` and `\ignorespacesafterend` commands (introduced in *Volume 1* [93, §10]) suppress any spaces following the start and end of the environment that may be introduced by a spurious EOL

9.3 The *probsoln* Package

character. The `\noindent` command (also introduced in Volume 1 [93, §10]) suppresses the paragraph indentation.

With this redefinition of the `solution` environment, the `exam` class's `\printanswers` command no longer has an effect. Instead the `probsoln` package's `\showanswers` command should be used.

Since the `textenum` environment is an inline list, its items may be placed inside a `tabular` environment, as is done in the example document below.

↑ Input

```
\documentclass[addpoints]{exam}

\usepackage{probsoln}

\showanswers

\renewenvironment{solution}%
{\par\noindent\textbf{\$solutionname}: \ignorespaces}%
{\ignorespacesafterend}

\renewcommand*\theenumi{\Alph{enumi}}

\begin{document}
```

9.3 The *probsoln* Package

```
\begin{center}\bfseries
Assignment~1\ifshowanswers\space (Solution Sheet)\fi
\end{center}

\begin{questions}
\question[1] What is the main drawback of ray guns?
\begin{onlysolution}
\begin{solution}
Overheating.
\end{solution}
\end{onlysolution}

\question[1] Which of the following is an ingredient of
mind-controlling cookies?
\begin{center}
\begin{textenum}
\begin{tabular}{ll}
\incorrectitem arsenic & \\
\incorrectitem cyanide \\
\incorrectitem curare & \\
\correctitem secret genetically modified sugar beet
\end{tabular}
\end{textenum}
\end{center}
\end{questions}
```

9.3 The *probsoln* Package

```
\end{textenum}
\end{center}
\end{questions}

\end{document}
```

↓ Input

The resulting text is:

↑ Output

Assignment 1 (Solution Sheet)

1. (1 point) What is the main drawback of ray guns?

Solution: Overheating.

2. (1 point) Which of the following is an ingredient of mind-controlling cookies?

A. arsenic B. cyanide

C. curare D. secret genetically modified sugar beet

↓ Output

9.3 The *probsoln* Package

You can [download](#) or [view](#) this example.

Thus far I haven't shown you anything that the exam class can't already do, so let's now look at how to define problems and their associated solutions for later use. A problem can be defined using the `defproblem` environment

```
\begin{defproblem}{\langle n \rangle}{\langle default args \rangle}{\langle label \rangle}{\langle option \rangle}  
{\text}  
\end{defproblem}
```

Definition

where `\langle text \rangle` may include the `onlyproblem` or `onlysolution` environments (or a combination of both). The first optional argument `\langle n \rangle` is the number of arguments this problem may take. Each argument may be referenced in `\langle text \rangle` using the standard `#1` etc method of referencing an argument. The second optional argument `\langle default args \rangle` are the default arguments to use with this problem when it is automatically used by `\thisproblem`, described below. (The default argument is ignored when the problem is referenced with `\useproblem`, which must have the arguments explicitly added.)

If `\langle n \rangle` is omitted, 0 is assumed and `\langle default args \rangle` should also be omitted. The final optional argument `\langle option \rangle` may be used to specify the `fragile` `boolean key` described earlier. If `\langle text \rangle` contains any instances of the `on-`

9.3 The *probsoln* Package

lyproblem or `onlysolution` environments, they will inherit the fragile state from `defproblem`.

The only mandatory argument is `<label>`, which is a label that uniquely identifies this problem so that it can later be referenced. As with all the other labelling systems described in this book, the label must not contain any `special characters`. The problem must be defined before use, typically either in the preamble or in an external `.tex` file.

EXAMPLE:

Here's a simple example that doesn't require any arguments:

```
\begin{defproblem}{raygun}
\begin{onlyproblem}
What is the main drawback of ray guns?%
\end{onlyproblem}%
\begin{onlysolution}
Overheating.% 
\end{onlysolution}
\end{defproblem}
```

↑ Input

↓ Input

9.3 The *probsoln* Package

Here's an example that requires one argument. This argument defaults to 2. Note that the braces `{ }` are required for each argument.

```
\begin{defproblem}[1][{2}]{diffsin}
\begin{onlyproblem}
Differentiate $f(x) = \sin(\#1x)$.
\end{onlyproblem}%
\begin{onlysolution}
$f'(x) = #1\cos(\#1x)$
\end{onlysolution}
\end{defproblem}
```

↑ Input

↓ Input

In both of the above examples, I'm assuming that the solutions will be printed later in the document, separate from the question, so I haven't bothered to use the `solution` environment, since the "Solution:" tag is now redundant. If, on the other hand, I want a solution sheet that displays the solution with its associated question, then it's better to remove the `onlyproblem` environment and add the `solution` environment inside the `onlysolution` environment:

9.3 The *probsoln* Package

↑ Input

```
\begin{defproblem}{raygun}
What is the main drawback of ray guns?
\begin{onlysolution}
\begin{solution}
Overheating.%\\
\end{solution}
\end{onlysolution}
\end{defproblem}
```

↓ Input

Since it's quite cumbersome having to write so many `\begin` and `\end` commands, the *probsoln* package provides a convenient shortcut command:

```
\newproblem[n][default args]{label}{problem}{solution}
```

Definition

This is equivalent to:

↑ Input

```
\begin{defproblem}[n][default args]{label}%
problem%
\begin{onlysolution}%

```

9.3 The *probsoln* Package

```
\begin{solution}%
<solution>%
\end{solution}%
\end{onlysolution}%
\end{defproblem}
```

↓ Input

There is also a starred version:

```
\newproblem*[<n>][<default args>]{<label>}{<problem>}
```

Definition

which is a shortcut for:

```
\begin{defproblem}<n>[<default args>]{<label>}%
<problem>%
\end{defproblem}
```

↑ Input

↓ Input

Note that both versions of `\newproblem` don't support verbatim, so if your problem contains verbatim text you must use the `defproblem` environment (with the `fragile` key set).

Once you have defined your problems, you can use them in your document. You can explicitly use a particular problem via the command:

9.3 The *probsoln* Package

`\useproblem[⟨dataset⟩]{⟨label⟩}{⟨arg₁⟩}…{⟨argₙ⟩}`

Definition

where $\langle \text{label} \rangle$ is the label uniquely identifying the problem. If the problem was defined to have arguments, the arguments must then follow the label. The optional argument $\langle \text{dataset} \rangle$ indicates the dataset in which the problem is stored. If omitted the default dataset is assumed.

EXAMPLE:

Given the definitions in the earlier example of the problems with the labels `raygun` and `diffsin`, these can now be used in the document:

```
\begin{enumerate}
\item \useproblem{raygun}

\item \useproblem{diffsin}{3}
\end{enumerate}
```

↑ Input

↓ Input

This is on the assumption that the problems were defined in the document preamble, which will automatically place them in the default dataset. Since the `diffsin` problem was defined to have one argument, that argument has to be provided.

9.3 The *probsoln* Package

If you define all your problems in external files, you can input each file using L^AT_EX's standard `\input` command, which will have the same effect as just defining all those problems within the document. Alternatively, you can load the problems into a particular dataset using one of the commands described below, where $\langle\text{dataset}\rangle$ is the name of the dataset and $\langle\text{filename}\rangle$ is the name of the file. If the dataset is omitted, the default dataset is assumed. Note that the dataset name mustn't contain any **special characters**.

`\loadallproblems[<dataset>]{<filename>}`

Definition

This loads all the problems defined in the given $\langle\text{filename}\rangle$ and adds them to the $\langle\text{dataset}\rangle$ indicated in the optional argument.

`\loadselectedproblems[<dataset>]{<labels>}{<filename>}`

Definition

This only defines the problems listed in the **comma-separated list** $\langle\text{labels}\rangle$. The other problems in $\langle\text{filename}\rangle$ are ignored.

`\loadexceptproblems[<dataset>]{<exception list>}{<filename>}`

Definition

This is the reverse of `\loadselectedproblems`. It only defines the problems that aren't listed in the **comma-separated list** $\langle\text{exception list}\rangle$.

9.3 The *probsoln* Package

`\loadrandomproblems[⟨dataset⟩]{⟨n⟩}{⟨filename⟩}` Definition

This loads $\langle n \rangle$ randomly selected problems defined in $\langle \text{filename} \rangle$ and adds them to the given dataset.

`\loadrandomexcept[⟨dataset⟩]{⟨n⟩}{⟨filename⟩}{⟨exception list⟩}` Definition

Similar to the previous command but discounts the problems listed in $\langle \text{exception list} \rangle$ when making the random selection.

Once you have loaded your problems, using one of the above commands, you can iterate through a dataset using:

`\foreachproblem[⟨dataset⟩]{⟨body⟩}` Definition

This does $\langle \text{body} \rangle$ at each iteration. Within $\langle \text{body} \rangle$ you can use

`\thisproblem` Definition

to use the current problem and

`\thisproblemlabel` Definition

to access the current problem label. Unlike `\useproblem`, you don't supply the problem arguments when you use `\thisproblem`. If the problem requires one or more arguments and no default arguments were provided when the problem was defined or the `usedefaultargs` package option

9.3 The *probsoln* Package

wasn't used, then you will be prompted for the arguments, which requires L^AT_EX to be run in interactive mode. (Interactive mode is when L^AT_EX stops on encountering an error and prompts you for a response.)

EXAMPLE:

To iterate through all problems in the default dataset:

```
\begin{enumerate}
\foreachproblem{\item\thisproblem}
\end{enumerate}
```

↑ Input

↓ Input

Assuming that you haven't switched on the solutions using `\showanswers` or the `answers` package option, this will just list all the problems. If you switch on the solutions, this will include the solutions but omit any text inside the `onlyproblem` environment.

There is also a similar command:

```
\foreachsolution[<dataset>]{<body>}
```

Definition

which behaves like `\foreachproblem` but only iterates over those problems that contain an `onlysolution` environment. (You must have first used the `problems` earlier in the document.) Note that you still need to switch on

9.3 The *probsoln* Package

the show solution flag using `\showanswers` or the `answers` package option if you want the solutions displayed.

`\foreachdataset{<cs>}{{<body>}}`

Definition

This iterates over all the defined datasets and does `<body>` at each iteration. Within `<body>` you can use the control sequence `<cs>` to access the current dataset name.

For example, to display all problems in all datasets:

```
\begin{enumerate}
\foreachdataset{\thisdataset}%
{%
  \foreachproblem[\thisdataset]{\item\thisproblem}%
}
\end{enumerate}
```

↑ Input

↓ Input

EXAMPLE:

Suppose I have some calculus problems defined in a file called `calculus.tex` and I have some linear algebra problems defined in a file called `linearalgebra.tex`, then in my document preamble I could, say, load five

9.3 The *probsoln* Package

randomly selected calculus problems and four randomly selected linear algebra problems using:

```
\loadrandomproblems[calculusproblems]{calculus}
\loadrandomproblems[linearalgebraproblems]{linearalgebra}
```

↑ Input

↓ Input

(The .tex extension may be omitted.) This creates two datasets with the labels calculusproblems and linearalgebraproblems. In my document, I could simply iterate over all datasets using `\foreachdataset` as described above.

```
\begin{enumerate}
\foreachdataset{\thisdataset}{%
{%
\foreachproblem[\thisdataset]{\item\thisproblem}{%
}}
\end{enumerate}
```

↑ Input

↓ Input

9.3 The *probsoln* Package

Alternatively, I might want to divide the document into sections for each topic:

```
\section{Calculus}
\begin{enumerate}
\foreachproblem[calculusproblems]{\item\thisproblem}
\end{enumerate}

\section{Linear Algebra}
\begin{enumerate}
\foreachproblem[linealgebraproblems]{\item\thisproblem}
\end{enumerate}
```

↑ Input

↓ Input

The seed used by the pseudo random number generator can be changed using:

```
\PSNrandseed{\langle n \rangle}
```

Definition

where $\langle n \rangle$ is a non-zero number to use as the seed. Random numbers can be generated using:

9.3 The *probsoln* Package

\PSNrandom{\langle register \rangle}{\langle n \rangle}

Definition

which generates a random integer between 1 and $\langle n \rangle$ (inclusive) and stores it in the given count *register* (see §2.1.3) or

\random{\langle counter \rangle}{\langle min \rangle}{\langle max \rangle}

Definition

which generates a random integer between $\langle min \rangle$ and $\langle max \rangle$ (inclusive) and stores it in the L^AT_EX *counter* whose name is *counter*. (See also §9.5 below.)

EXAMPLE:

Recall the earlier *diffsin* problem. Suppose that instead of defining the problem to have an argument for the factor, the factor is randomly selected instead. First a new register needs to be defined:

\newcount\myrandarg

Input

Now the problem can be defined:

```
\begin{defproblem}{randdифfsin}
\PSNrandom{\myrandarg}{10}%
Differentiate $f(x) = \sin(\the\myrandarg x)$.
```

↑ Input

9.3 The *probsoln* Package

```
\begin{onlysolution}
$f'(x) = \the\myrandarg\cos(\the\myrandarg x)$
\end{onlysolution}
\end{defproblem}
```

↓ Input

If the random number seed is set to the current year:

```
\PSNrandseed\year
```

Input

then this problem will appear differently if the same document is rebuilt on different years.

 If you want the solutions to appear in a different location to the questions (for example, at the end of the document) you'll need to store the randomly generated number to prevent a different number from being generated in the solution section. For example:

```
\begin{defproblem}{randdiffsin}
\ifundefined{randdiffsinarg}
{%
\PSNrandom{\myrandarg}{10}%
\xdef\randdiffsinarg{\the\myrandarg}% globally store
}
```

↑ Input

9.3 The *probsoln* Package

```
}

{ } %

Differentiate $f(x) = \sin(\randdiffsinarg x)$.

\begin{onlysolution}
$f'(x) = \randdiffsinarg \cos(\randdiffsinarg x)$
\end{onlysolution}
\end{defproblem}
```

↓ Input

(See also §9.5.)

You can iterate over $\langle n \rangle$ randomly selected items in a [comma-separated list](#) using:

`\doforrandN{\langle n \rangle}{\langle cs \rangle}{\langle list \rangle}{\langle body \rangle}`

[Definition](#)

This performs $\langle body \rangle$ at each iteration where $\langle cs \rangle$ is a control sequence set to the currently selected item.

EXAMPLE:

Suppose you have four files called `file1.tex`, `file2.tex`, `file3.tex` and `file4.tex` that all contain problem definitions, but you only want to load the problems defined in two of those files selected at random:

9.3 The *probsoln* Package

↑ Input

```
\doforrandN  
{2}% randomly select 2 items from the list  
\thisfile% command in which to store the current item  
{file1,file2,file3,file4}% the list  
\loadallproblems{\thisfile}
```

↓ Input

EXAMPLE:

In this example, only one random selection is made and the selected item is saved for later use:

↑ Input

```
\doforrandN{1}{\thisitem}{cow,duck,chicken}  
 {\global\let\thesubject\thisitem}%  
\doforrandN{1}{\thisitem}{road,field,river}  
 {\global\let\theobject\thisitem}%  
Why did the \thesubject\_\_cross the \theobject?  
\begin{onlysolution}  
 \begin{solution}
```

9.3 The *probsoln* Package

```
So that the \thesubject\_\_could get to the other side of the  
\theobject.  
\end{solution}  
\end{onlysolution}
```

↓ Input

The previous warning also applies here if you intend to display the solutions in another part of the document. In this case, a similar approach can be used:

```
\begin{defproblem}{whycross}  
\ifdef{\thesubject}  
{% already defined  
{%  
  \doforrandN{1}{\thisitem}{cow,duck,chicken}  
    {\global\let\thesubject\thisitem}%  
  \doforrandN{1}{\thisitem}{road,field,river}  
    {\global\let\theobject\thisitem}%  
}%">  
\begin{onlyproblem}  
Why did the \thesubject\_\_cross the \theobject?
```

↑ Input

9.3 The *probsoln* Package

```
\end{onlyproblem}
\begin{onlysolution}
So that the \thesubject\_\_ could get to the other side of the
\theobject.
\end{onlysolution}
\end{defproblem}
```

↓ Input

This uses the `\ifdef` command provided by the `etoolbox` package described in §2.1.1.

Note that the `pgfmath` package also provides pseudo-random commands, which you may prefer to use. Some of these are described in §9.5.

EXAMPLE 45. RANDOMLY SELECTING PROBLEMS

In this example, I have a number of files containing problem definitions:

1. `mth101.tex`

This file contains 10 easy differentiation problems in the form:

```
\begin{defproblem}{diff:sinx/x}
\begin{onlyproblem}%

```

↑ Input

9.3 The *probsoln* Package

```
$y = \frac{\sin x}{x}.
```

```
\end{onlyproblem}
```

```
\begin{onlysolution}%
```

```
\[\frac{dy}{dx} = \frac{\cos x}{x} - \frac{\sin x}{x^2}\]
```

```
\end{onlysolution}%
```

```
\end{defproblem}
```

↓ Input

You can [download](#) the complete file.

2. `problems-1stprinciples.tex`

This file contains 5 differentiation from first principle problems in the form:

```
\begin{defproblem}{dfp:cons}%
```

```
\begin{onlyproblem}%
```

```
Differentiate from first principles $f(x) = c$ where
```

```
$c$ is a constant.%
```

```
\end{onlyproblem}%
```

↑ Input

9.3 The *probsoln* Package

```
\begin{onlysolution}%
\begin{align*}
\frac{df}{dx} &=
\lim_{\Delta x \rightarrow 0} \frac{c - c}{\Delta x} \\
&= \lim_{\Delta x \rightarrow 0} \frac{0}{0} \\
&= 0
\end{align*}
\end{onlysolution}%
\end{defproblem}
```

↓ Input

You can [download](#) the complete file.

I can now write a document that randomly selects 3 problems from the first file and 1 problem from the second file. The questions are listed first and the solutions later:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{probsoln}
```

↑ Input

9.3 The *probsoln* Package

```
\loadrandomproblems{3}{mth101}
\loadrandomproblems{1}{problems-1stprinciples}

\begin{document}
\section{Differentiation Problems}
\begin{enumerate}
\foreachproblem{\item\thisproblem}
\end{enumerate}

\section{Solutions}
\showanswers
\begin{enumerate}
\foreachsolution{\item\thisproblem}
\end{enumerate}
\end{document}
```

↓ Input

You can [download](#) or [view](#) this example document.

9.4 Using the `datatool` Package for Exams or Assignment Sheets

Since the `datatool` package and the `datatooltk` application have already been described in this book, it's worth mentioning that they can also be used to store a database of problems and their associated solutions. This can be done by creating a database with a label field, a question field and an answer field. Other fields can also be added to store, for example, the topic or level of difficulty.

If you already have a file containing `probsoln` problem definitions, `datatooltk` can convert it to a `datatool` database.¹ For example, the `mth101.tex` file from [Example 45](#) can be imported either using the `--probsoln` command line option or the `File→Import→Import probsoln` file menu item in the [GUI](#) mode. [Figure 9.1](#) shows the `mth101.tex` file imported into `datatooltk`. Since [L^AT_EX](#) is used to assist the conversion, the “pretty-printing” of the code has unfortunately been lost, but this won't affect the typeset output. (This also happens if you use `\DTLsaverawdb` or `\DTLprotectedsaverawdb`.)

The import process has created three fields: `Label`, `Question` and `Answer`. Extra fields can be added using the `Edit→Column→Insert Column After` menu item. For example, in [Figure 9.2](#), I've added a new integer field called

¹You can't export back to the `probsoln` format.

9.4 Using the *datatool* Package for Exams or Assignment Sheets

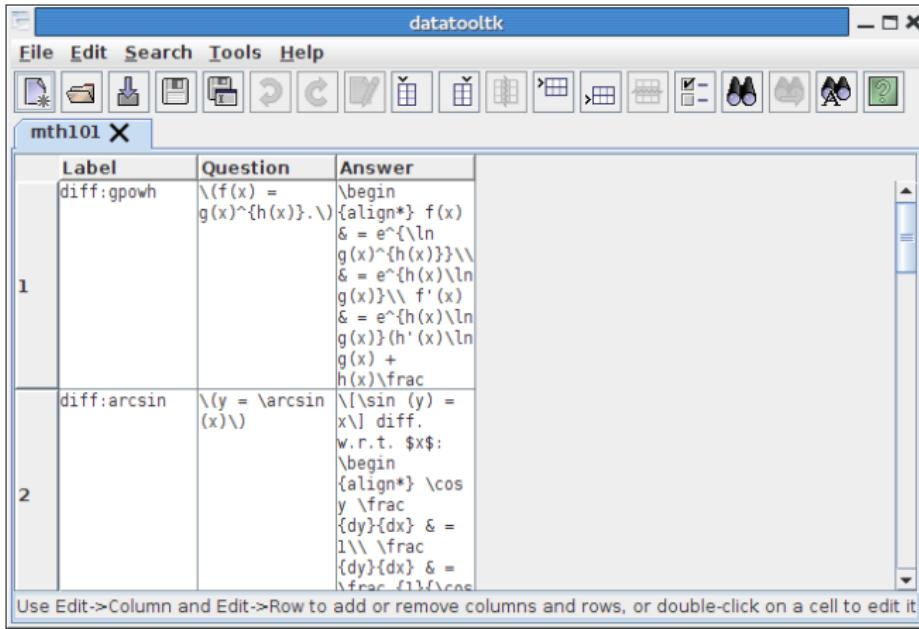


Figure 9.1 Importing a `probsoln` Dataset into `datatooltk`

9.4 Using the `datatool` Package for Exams or Assignment Sheets

Level, where a value of 1 indicates easy, 2 indicates medium difficulty and 3 indicates hard. This database can then be saved as, say, `mth101.dbtex` and loaded into a document using `\DTLloaddbtx`, as described in §2.2.2. You can add other columns as well, such as a topic.

⚠ Note that `datatool` has a drawback that `probsoln` doesn't have, and that is the lack of support for verbatim. You can, however, use `\lstinputlisting` (provided by the `listings` package [34], described in Volume 2 [96, §4.5]) or `\verbatiminput` (provided by the `verbatim` package [85]).

A new boolean variable can be defined using:

`\newboolean{<name>}`

Definition

defined by the `ifthen` package, or

`\newbool{<name>}`

Definition

defined by the `etoolbox` package, where `<name>` is the name of the variable. (Note that `<name>` is not a control sequence.) The state can be set using:

`\setboolean{<name>}{{<state>}}`

Definition

defined by the `ifthen` package, or

`\setbool{<name>}{{<state>}}`

Definition

9.4 Using the *datatool* Package for Exams or Assignment Sheets

The screenshot shows the *datatooltk* application window. The menu bar includes File, Edit, Search, Tools, and Help. Below the menu is a toolbar with various icons. The main area displays a table titled "mth101 *". The table has four columns: Label, Question, Answer, and Level. The "Label" column contains identifiers like "diff:gpowh" and "diff:arcsin". The "Question" and "Answer" columns contain LaTeX code snippets. A new "Level" column is added to the right, with values 3 and 2 respectively for the two rows.

Label	Question	Answer	Level
1	diff:gpowh	\begin{aligned} f(x) &= g(x)^{h(x)} \\ &= e^{\ln(g(x)^{h(x)})} \\ &= e^{h(x)\ln(g(x))} \end{aligned}	3
2	diff:arcsin	\begin{aligned} y &= \arcsin(x) \\ \text{diff. w.r.t. } x &: \end{aligned}	2

Figure 9.2 New Level Column Added

9.4 Using the `datatool` Package for Exams or Assignment Sheets

defined by the `etoolbox` package, where `<state>` may be either `true` or `false`. With the `etoolbox` package, you can also use:

`\boolfalse{<name>}`

Definition

to set the state to `false` or

`\booltrue{<name>}`

Definition

to set the state to `true`.

The variable's state can be tested using:

`\ifthenelse{\boolean{<name>}}{<true>}{<false>}`

Definition

defined by the `ifthen` package, or

`\ifbool{<name>}{<true>}{<false>}`

Definition

defined by the `etoolbox` package.

Note that `\newboolean` and `\newbool` both use the same underlying `\def` command to define a conditional so they have the same effect. The `etoolbox` `\setbool` can be prefixed with `\global` but `ifthen`'s `\setboolean` can't.

It's therefore possible to define your own boolean flag that determines whether or not the solutions should be displayed.

9.4 Using the `datatool` Package for Exams or Assignment Sheets

EXAMPLE 46. CREATING A PROBLEM SHEET USING `datatool`

Returning to the database shown in Figure 9.2. Suppose that database is saved as `mth101.dbtex`. Now it can be loaded and iterated over to display all the questions:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

\newbool{showanswers}
\booltrue{showanswers}

\DTLloaddbtx{\problemDB}{mth101.dbtex}

\begin{document}
\begin{center}\bfseries\Large
Assignment~1\ifbool{showanswers}{}{(Solution Sheet)}\ifbool{showanswers}{}{\vphantom{A}}
\end{center}

\begin{enumerate}
```

↑ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\DTLforeach*{\problemDB}{\Label=Label,\Question=Question,\Answer=Answer}%
{%
  \item \Question
  \ifbool{showanswers}{\par\textbf{Solution: } \Answer}{}%
}
\end{enumerate}

\end{document}
```

↓ Input

You can [download](#) or [view](#) this example document.

Alternatively, you could gather all the solutions at the end of the document:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

\newbool{showanswers}
```

↑ Input

9.4 Using the *datatool* Package for Exams or Assignment Sheets

```
\booltrue{showanswers}

\DTLloaddbtx{\problemDB}{mth101.dbtex}

\begin{document}
\begin{center}\bfseries\Large
Assignment~1
\end{center}

\begin{enumerate}
\DTLforeach*{\problemDB}
  {\Label=Label,\Question=Question}%
{%
  \item \Question
}
\end{enumerate}

\ifbool{showanswers}
{%
  \section{Solutions}

\begin{enumerate}
```

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\DTLforeach*{\problemDB}{\Label=Label,\Answer=Answer}%
{%
  \item \Answer
}
\end{enumerate}

}{}%
\end{document}
```

↓ Input

You can, of course, use the `exam` class or `probsoln` package with `datatool`. That way you don't need to define your own boolean variable.

It may, however, be that you only want a random selection of the questions from the database. While this could be done within the document using commands provided by the `datatool` package, it's more efficient to do this using `datatooltk`. That way, the random selection only needs to be done once per problem sheet (possibly repeated after any modifications to the database) which reduces the time taken for `TeX` to compile the document. The `datatooltk` has a number of command line options that can help with this:

9.4 Using the `datatool` Package for Exams or Assignment Sheets

- **--shuffle**

Shuffle the rows in the database.
- **--seed <number>**

Set the random generator seed to <number>.
- **--shuffle-iterations <number>**

Sets the number of iterations performed in the shuffle to <number>
- **--truncate <number>**

Truncate the database to the first <number> rows. (This option is always performed after the shuffle option, regardless of the option order.)
- **--filter <key> <operator> <value>**

Adds a filter. This option may be used multiple times. Here <key> is the column label used by the filter. The <operator> may be one of: `eq` (equals), `ne` (does not equal), `le` (is less than or equal to), `lt` (is less than), `ge` (is greater than or equal to), `gt` (is greater than) or `regex` (matches the regular expression). In the last case, <value> should be a regular expression as used by `java.util.regex.Pattern`. In the

9.4 Using the `datatool` Package for Exams or Assignment Sheets

other cases, `<value>` may be an integer, real number or string. If the datatype for the column identified by `<key>` is numerical and `<value>` is also numerical, then a numerical comparison is used, otherwise a string comparison is used. For example, `--filter Level le 2` indicates that the filter should return a true value for any row where the value in the `Level` column is less than or equal to 2.

Filtering is always applied after shuffling and before truncating (if either of those options have been specified).

- `--filter-and`

The default action in the event of multiple `--filter` options is to apply a logical “or”. The `--filter-and` changes this behaviour to apply a logical “and” to all the filter results instead. For example, suppose the database also has a column labelled `Topic` and you want to select five easy questions from the topic “Algebra”, then you need a logical “and”:

```
datatooltk --in mth101.dbtex --shuffle --filter-and  
--filter Level eq 1 --filter Topic eq Algebra  
--truncate --output problems.dbtex
```

Shell

9.4 Using the `datatool` Package for Exams or Assignment Sheets

- `--filter-exclude`

When applying any filters, the `--filter-exclude` option will cause any matching rows to be excluded. (The default behaviour is to exclude non-matching rows.)

- `--merge <col-label> <filename>`

Merges the loaded database with the database in the file whose name is given by `<filename>`. The merge is performed by merging each row in `<filename>` with the row in the database where the column given by the label `<col-label>` has the same value as the column with the same label in `<filename>`. If no match is found, a new row is added.

With a combination of these options, it's possible to create a database file (called, say, `problems.dbtex`) that only contains a random subset of the complete database.

EXAMPLES:

1. Select five questions (of any level) at random:

```
datatooltk --in mth101.dbtex --shuffle --truncate 5  
--output problems.dbtex
```

Shell

9.4 Using the *datatool* Package for Exams or Assignment Sheets

2. Select two level 1 questions at random:

```
datatooltk --in mth101.dbtex --shuffle  
--filter Level eq 1 --truncate 5  
--output problems.dbtex
```

Shell

3. Select four non-easy questions at random with the seed set to 2014:

```
datatooltk --in mth101.dbtex --shuffle --seed 2014  
--filter Level ne 1 --truncate 4  
--output problems.dbtex
```

Shell

The document from [Example 46](#) just needs one line changed, and that's the line that loads the database:

```
\DTLloaddbtex{\problemDB}{problems.dbtex}
```

Input

Alternatively, if you want, say, four level 1 questions, two level 2 questions and one level 3 question, you can create three separate databases:

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
datatooltk --in mth101.dbtex --shuffle --filter Level eq 1  
--truncate 4 --output problems1.dbtex  
datatooltk --in mth101.dbtex --shuffle --filter Level eq 2  
--truncate 2 --output problems2.dbtex  
datatooltk --in mth101.dbtex --shuffle --filter Level eq 3  
--truncate 1 --output problems3.dbtex
```

Shell

Now you need to load all three databases into your document:

```
\DTLloaddbtex{\problemDBi}{problems1.dbtex}  
\DTLloaddbtex{\problemDBii}{problems2.dbtex}  
\DTLloaddbtex{\problemDBiii}{problems3.dbtex}
```

↑ Input

↓ Input

and iterate over each of them:

```
\begin{enumerate}  
\DTLforeach*{\problemDBi}{  
{\Label=Label,\Question=Question,\Answer=Answer}}%
```

↑ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
{%
  \item \Question
  \ifbool{showanswers}{\par\textbf{Solution: }}{\Answer}{}%
}
\DTLforeach*{\problemDBii}
  {\Label=Label,\Question=Question,\Answer=Answer}%
{%
  \item \Question
  \ifbool{showanswers}{\par\textbf{Solution: }}{\Answer}{}%
}
\DTLforeach*{\problemDBiii}
  {\Label=Label,\Question=Question,\Answer=Answer}%
{%
  \item \Question
  \ifbool{showanswers}{\par\textbf{Solution: }}{\Answer}{}%
}
\end{enumerate}
```

↓ Input

If you do intend to do this, I suggest you define a command to perform these iterations. For example:

↑ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\newcommand{\doquestions}[1]{%
  \DTLforeach*{#1}{%
    {\Label=Label,\Question=Question,\Answer=Answer}%
    {%
      \item \Question
      \ifbool{showanswers}{\par\textbf{Solution: }\Answer}{}%
    }%
  }%
}
```

↓ Input

If the original database contains, say, two hundred problems, using `datatool` in this way can significantly speed up the document build. Each year you can run the `datatool` commands with a different random generator seed to produce a new assignment sheet or exam paper.

If you prefer to store your problems in a `SQL` database, you can perform the random selection with the `SELECT` statement. For example, if the problems are stored in a table called `calculus` within a database called `mth101`, then you can select, say, five questions at random using:

```
datatool --output problems.dbtex --sqluser username
--sqldb mth101 --sql "SELECT * FROM calculus ORDER BY RAND()
LIMIT 5"
```

Shell

9.4 Using the `datatool` Package for Exams or Assignment Sheets

What if you don't want to select any problems that appeared in the exam paper or assignment sheet in, say, the previous two years? You could add a year column to the original complete database, but this can be tiresome and prone to error if done manually. It could possibly be done by the L^AT_EX document, but this would require loading the entire database and saving it using `\DTLsaverawdb`, which means it's pointless using the `datatooltk` options described above and, as noted earlier, you'd lose any pretty-printing in the code.

Instead, I think it's more practical to keep a separate database containing just the problem labels and the year that problem was selected. This database can be updated by the document, but since any problems that haven't been used in the past two years can be discarded, this database is much smaller than the original database. Let's call this database file, say, `mth101-years.dbtex`. On the first year, this file won't exist. Recall from [Example 33](#) the `\InputIfFileExists` command. If the file doesn't exist, a new database can be created using:

`\DTLnewdb{\langle db-name\rangle}`

Definition

where `\langle db-name\rangle` is the database name.

EXAMPLE:

↑ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\InputIfFileExists{mth101-years.dbtex}
{ }% file exists
{\DTLnewdb{mth101-years}}% file doesn't exist
```

↓ Input

While the main database is iterated over, each question label can be added to the `mth101-years` database with the current year. To add data, you first need to add a new row to the database using:

```
\DTLnewrow{<db-name>}
```

Definition

and then you can add the entries for that row using:

```
\DTLnewdbentry{<db-name>}{<col-label>}{{<value>}}
```

Definition

where `<col-label>` is the column label and `<value>` is the value for that column. By default, the value isn't expanded. To change this, you first need to use the command:

```
\dtlexpandnewvalue
```

Definition

EXAMPLE 47. RANDOMLY SELECTING PROBLEMS NOT USED IN THE PAST TWO YEARS

(This exercise assumes that the current year is 2014.) Adapting the earlier code from [Example 46](#):

9.4 Using the *datatool* Package for Exams or Assignment Sheets

↑ Input

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

\newbool{showanswers}
\booltrue{showanswers}

\DTLloaddbtx{\problemDB}{mth101.dbtex}

\InputIfFileExists{mth101-years.dbtex}
{}% file exists
{\DTLnewdb{mth101-years}}% file doesn't exist

\begin{document}
\begin{center}\bfseries\Large
Assignment~1\ifbool{showanswers}{}{(Solution Sheet)}\ifbool{showanswers}{}{1}
\end{center}

\dtlexpandnewvalue
```

9.4 Using the *datatool* Package for Exams or Assignment Sheets

```
\begin{enumerate}
\DTLforeach*{\problemDB}{%
  {\Label=Label,\Question=Question,\Answer=Answer}%
{%
  \item \Question
  % add this label to the new database:
  \DTLnewrow{mth101-years}%
  % add a new row
  \DTLnewdbentry{mth101-years}{Label}{\Label}%
  \DTLnewdbentry{mth101-years}{Year}{\number\year}%
  % print the solution if this is the answer sheet:
  \ifbool{showanswers}{\par\textrm{\bf{Solution: }}\Answer}{}%
}
\end{enumerate}
```

↓ Input

At the end of the document, the database needs to be saved:

```
\DTLsaverawdb{mth101-years}{mth101-years.dbtex}
\end{document}
```

↑ Input

↓ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

(You can [download](#) or [view](#) this document.)

The call to `datatooltk` can use the `--merge` command line option. For example, to randomly select five problems:

```
datatooltk --in mth101.dbtex --merge Label mth101-years.dbtex
--shuffle --filter-and --filter Year ne 2013
--filter Year ne 2012 --truncate 5 --output problems.dbtex
```

Shell

If the `mth101` database doesn't need editing, this call only really needs to be done once a year. However, if you edit the database by removing, adding or swapping rows, you may end up with a different selection, and labels that are no longer selected will still be assigned to the current year. For example, suppose `diff:arccsin` was selected for this year, but then you add another problem to `mth101.dbtex` so that now `diff:arccsin` is no longer selected, but it's still listed in `mth101-years.dbtex` as having been selected this year. You can fix this using:

```
datatooltk --in mth101-years.dbtex --filter Year eq 2013
--filter Year eq 2012 --output mth101-years.dbtex
```

Shell

This also has the advantage of removing any problems from pre-2012, which trims down the database.

9.4 Using the `datatool` Package for Exams or Assignment Sheets

If you use `make` on a Unix-like system, the Makefile could look something like:

```
CURRYEAR:=$(shell date +%Y)
LASTYEAR:=$(shell expr $(CURRYEAR) - 1)
YEARBEFORE:=$(shell expr $(CURRYEAR) - 2)

assignmentsheet1.pdf      : assignmentsheet1.tex problems.dbtex
                           pdflatex assignmentsheet1

problems.dbtex  : mth101.dbtex
                  datatooltk --in mth101.dbtex \
                  --merge Label mth101-years.dbtex \
                  --shuffle \
                  --filter-and \
                  --filter Year ne $(LASTYEAR) \
                  --filter Year ne $(YEARBEFORE) \
                  --truncate 5 \
                  --output problems.dbtex

update        :
                  datatooltk --in mth101-years.dbtex \
                  --filter Year eq $(LASTYEAR) \
```

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
--filter Year eq $(YEARBEFORE) \
--output mth101-years.dbtex
```

Now, at the start of each year (or after altering the structure of the database in `mth101.dbtex`) you can use

```
make update
```

Shell

to trim `mth101-years.dbtex` to just the entries for the previous two years. (There's probably a more efficient way of writing this Makefile, but a discussion of the `make` utility is beyond the scope of this book. If you want to copy the above code, remember to use the TAB character in the appropriate places. Alternatively, you can [download](#) the file from the examples directory.)

Note that the `--merge` option will be ignored if the file to be merged doesn't exist. (Just a warning message will be displayed on the standard error stream.) This means that the `problems.dbtex` target will work on the first instance, even though the `mth101-years.dbtex` file doesn't exist.

Recall the `\marginpar` command from [Exercise 21](#). This can be used to, say, display the number of points for a question in the margin. For example, if all questions are worth 20 points, then within the body of `\DTLforeach` the number of points can be inserted into the margin:

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\item \marginpar{(20 points)}\Question
```

Input

Although it may be better to define a command called, say, `\points` to make it easier to customize. For example, in the preamble:

```
\newcommand*{\points}[1]{%
  \marginpar{(#1 points)}%
}
```

↑ Input

↓ Input

Then the body code of `\DTLforeach` can be simplified:

```
\item \points{20}\Question
```

Input

Now you just need to modify the definition of `\points` if you want to change the way the points are displayed. For example, if the argument of `\points` is always an integer, you could check for a single point and change “points” to “point”:

```
\newcommand*{\points}[1]{%
```

↑ Input

9.4 Using the *datatool* Package for Exams or Assignment Sheets

```
\marginpar{(#1
\ifnum#1=1\relax
    point%
\else
    points%
\fi)}%
}
```

↓ Input

If the argument may be a decimal number, the *datatool* package provides the command:

```
\dtlifnumeq{<number 1>}{{<number 2>}}{<true>}{<false>}
```

Definition

which can be used with decimal numbers. For example:

```
\newcommand*\points[1]{%
\marginpar{(#1
\dtlifnumeq{#1}{1}{point}{points})}}%
```

↑ Input

↓ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

Perhaps the points should depend on the difficulty level. For example, 5 points for a level 1 question, 10 points for a level 2 question and 20 points for a level 3 question. The `\ifcase` command described in §7.3 can be used to check the level:

```
\item
  \ifcase\Level
  \or
    \points{5}%
  \or
    \points{10}%
  \or
    \points{20}%
  \fi
\Question
```

↑ Input

↓ Input

Again, you can define a command that will simplify the document code:

```
\newcommand*{\PointsForLevel}[1]{%
```

↑ Input

9.4 Using the `datatool` Package for Exams or Assignment Sheets

```
\ifcase#1
\or
  \points{5}%
\or
  \points{10}%
\or
  \points{20}%
\fi
}
```

↓ Input

Now the code in the loop is:

```
\item \PointsForLevel{\Level}\Question
```

Input

EXERCISE 26. CREATING AN ASSIGNMENT SHEET WITH THE `datatool` PACKAGE

The `exercises` directory that comes with this book has a database called `mth102.dbtex` (shown in Figure 9.3). You can [download](#) this file or create your own. This database is an amalgamation of the two databases from Example 45 with an extra column labelled “Topic”. The topics are set to either “Basic” or “Theory”. The questions taken from the `problems-1stprinciples` database have all been given a value of 3 for the level. Create an assign-

9.4 Using the *datatool* Package for Exams or Assignment Sheets

ment sheet (or exam paper) that has the questions randomly selected from the `mth102` database. There should be two Level 1 questions from the “Basics” topic, one Level 2 question from the “Basics” topic and one Level 3 question from the “Theory” topic. Each question should have the points displayed, using the above point allocation scheme.

FOR THE MORE ADVENTUROUS:

Adjust the `\points` command so that it keeps a running total. This total should ideally occur at the start of the document, but as the value isn’t known until the end of the document, the information needs to be written to the auxiliary (`.aux`) file. L^AT_EX provides the command:

`\protected@write{\output stream}{\init code}{\text{}}`

Definition

which will write `\text{}` to the file identified by `\output stream`. The second argument, `\init code`, is provided for any initialisation that needs to be done prior to writing the text. The output stream for the document’s auxiliary file is identified by the command `\@auxout`. You’ll need to wrap the point total up in a command that can be used to reference the total at the start of the next run. Remember to use `\protect` in `\text{}` to prevent expansion of this helper command.

You can [download](#) or [view](#) a solution to this exercise.

9.4 Using the *datatool* Package for Exams or Assignment Sheets

Figure 9.3 The `mth102` Database

9.5 ■ Random Numbers

The previous sections have looked at randomly selecting problems from a database, but it may be that you want to generate questions that use random numbers (for example, as coefficients) to make a slightly different problem each year. In addition to the random number command `\PSNrandom` provided by `probsoln`, both the `fp` and `pgfmath` packages provide a way of randomly generating numbers. In the case of the `fp` package, you can generate a random number between 0 and 1 using

`\FPrandom{\langle cs\rangle}`

Definition

where $\langle cs \rangle$ is a control sequence in which to store the random number. The random number generator seed is set using

`\FPseed=(number)`

Definition

(`\FPseed` is a count register.) For example

`\FPseed=\year`

Input

will set the seed to the current year.

The `pgfmath` package provides

9.5 Random Numbers

`\pgfmathparse{\langle expression \rangle}`

Definition

which parses the given mathematical expression and sets `\pgfmathresult` to the result. There are a number of functions that may be used within `\langle expression \rangle` (see the pgf user guide [102] for further details) but the random generator functions are

`rnd`

Definition

which generates a number between 0 and 1,

`rand`

Definition

which generates a number between -1 and 1, and

`random(\langle x \rangle, \langle y \rangle)`

Definition

which generates a random integer between `\langle x \rangle` and `\langle y \rangle`, if both are present, or a random integer between 1 and `\langle x \rangle` if only `\langle x \rangle` is present:

`random(\langle x \rangle)`

Definition

or a random number between 0 and 1 if no arguments are present:

`random()`

Definition

The random number seed can be set using:

9.5 Random Numbers

```
\pgfmathsetseed{<n>}
```

Definition

where $\langle n \rangle$ is an integer. For example:

```
Year: \the\year.  
\pgfmathsetseed{\year}  
\pgfmathparse{random(2,10)}  
Random number: \pgfmathresult.
```

↑ Input

↓ Input

produces:

Year: 2015. Random number: 5.

Output

Additionally, the pgfmath package also provides:

```
\pgfmathrandominteger{<cs>}{{<minimum>}}{{<maximum>}}
```

Definition

which defines the control sequence $\langle cs \rangle$ to be a pseudo-randomly generated integer between $\langle minimum \rangle$ and $\langle maximum \rangle$ (inclusive). You can also define a list from which you want to randomly select an item. First you need to define the list using:

9.5 Random Numbers

`\pgfmathdeclarerandomlist{<list name>}{{<item 1>}{<item 2>}...}`

Definition

where *<list name>* is the name of the list and *<item 1>*, *<item 2>* etc are the list items. (Note that this list isn't a [comma-separated list](#). Each item is in braces like an argument.)

Once the list has been defined you can randomly select an item using:

`\pgfmathrandomitem{<cs>}{<list name>}`

Definition

where *<list name>* identifies the list. The result can then be accessed using the supplied control sequence *<cs>*.

EXAMPLE:

```
% define list
\pgfmathdeclarerandomlist{projects}%
% list items
{ray-guns}% first item
{mind-controlling cookies}% second item
{exploding chocolates}% third item
{telepathic cakes}% fourth item
}
```

↑ Input

9.5 Random Numbers

```
% randomly select an item from the list  
\pgfmathrandomitem{\thisproject}{projects}
```

What are the advantages and drawbacks of \thisproject?

↓ Input

⚠ As mentioned earlier, take care if you are using a mechanism that first displays questions and later (for example, at the end of the document) displays the solutions as this can cause a different randomly generated value in the solution. As before, I recommend that the question part globally defines a command that stores the randomly generated value which can later be accessed in the solution.

EXAMPLE 48. RANDOM SELECTION WITH pgfmath AND probsoln

(Recall the commands `\ifundefined`, `\global` and `\let` from §2.1.1.)

```
\documentclass{article}
```

```
\usepackage{pgfmath}  
\usepackage{probsoln}
```

```
% set random seed
```

↑ Input

9.5 Random Numbers

```
\pgfmathsetseed{\year}

\begin{defproblem}{easy.diff}%
\ifundefined{easydiffcoeff}
{%
  \pgfmathrandominteger{\easydiffcoeff}{2}{10}% random coefficient
  \global\let\easydiffcoeff\easydiffcoeff % make it global
}
{}% already been defined
\begin{onlyproblem}
% question
Differentiate with respect to $x$:
\[
y = \sin(\easydiffcoeff x)
\]
\end{onlyproblem}
\begin{onlysolution}
% solution
$ y' = \easydiffcoeff \cos(\easydiffcoeff x) $ 
\end{onlysolution}
\end{defproblem}
```

9.5 Random Numbers

```
\begin{document}

\section{Questions}

\begin{enumerate}
\foreachproblem{\item\thisproblem}
\end{enumerate}

\showanswers
\section{Solutions}
\begin{enumerate}
\foreachsolution{\item\thisproblem}
\end{enumerate}

\end{document}
```

↓ Input

This produces (where the year is 2014) the result shown in [Figure 9.4](#). You can [download](#) or [view](#) this example.

1 Questions

1. Differentiate with respect to x :

$$y = \sin(8x)$$

2 Solutions

1. $y' = 8 \cos(8x)$

Figure 9.4 Random Selection with pgfmath and probsoln

10. ■ BUSINESS CARDS, FLYERS AND LEAFLETS

There are only a few packages on [CTAN](#) for typesetting business cards listed under the [file-card topic](#), and only one of them is for $\text{\LaTeX}2_{\mathcal{E}}$ (rather than the old $\text{\LaTeX}2.09$) and that's *bizcard* [43]. At the time of writing, the current version is 1.1 dated 1999-09-04. It's in both MiK \TeX and \TeX Live, but is only suitable for the US 76.2 mm \times 50.8 mm (2 in \times 3.5 in) card size.

The [labels topic](#) provides some more options, but discounting Plain \TeX and $\text{\LaTeX}2.09$ options and also discounting packages that aren't in both MiK \TeX and \TeX Live, only three remain: *jlabels* [104] (for making letter-sized pages of labels), *labels* [68] (designed for sheets of Avery 5360 sticky labels, which could possibly be adapted for business cards) and *ticket* (for making labels, visiting-cards and pins). As far as I can tell, all these packages, with the exception of *ticket*, appear to be designed for letter paper.

The *ticket* package can be used with either letter or A4 paper (or any other size, if required). You just need to set up the paper margins, the size of each card, the distance between the cards and the number of rows and columns required to position the cards on the page. This package uses \LaTeX 's [picture environment](#), which is an unsophisticated but platform-

independent drawing environment.

The [layout topic](#) has a much greater list. Some of these are general purpose packages for positioning text or graphics on the page. The only one that seems to be specifically designed for leaflets is the `leaflet` class (version 1.0e 2013-11-06, at the time of writing) which can be used to create z-fold leaflets. The general purpose positioning packages include the `flowfram` package, which has an optional helper [GUI](#) application called `flowframtk`.

There don't appear to be any classes or packages listed on [CTAN](#) that specifically deal with single-paged flyers, but that's hardly surprising given the lack of structure in such a document. A flyer would typically need a graphics environment, such as the `picture` environment, to insert the required text and graphics, although a very simple text-only flyer could be created using the `article` class with an empty page style.

It's sometimes necessary to include bar codes, such as QR codes, in leaflets or flyers. There are a number of bar code packages listed in the [barcode topic](#). For brevity, only `pst-barcode` is discussed.

The rest of this chapter is arranged as follows:

- §10.1** Describes how to use the `picture` environment (which is needed for the subsequent sections on using the `ticket` package and `leaflet` class).
- §10.2** Describes how to use the `ticket` package to create business cards.
- §10.3** Describes how to use the `leaflet` class.

10.1 The picture Environment

§10.4 Describes how to use the `pst-barcode` class.

§10.5 Describes the `flowfram` package and the `flowframtk` application.

10.1 ■ The picture Environment

The `picture` environment is defined in the L^AT_EX kernel, so you don't need to load any packages to use it. However, it's rather restrictive as to what you can actually draw.

[FAQ: Drawing
with T_EX]

```
\begin{picture}(<width>,<height>)(<llx>,<llx>)

\end{picture}
```

Definition

This has a different syntax to most of the standard environments as the arguments are placed in parentheses rather than curly braces. Another unusual aspect is that the second argument ($<llx>$, $<llx>$) is optional, even though it isn't delimited by square brackets.

The first argument ($<width>$, $<height>$) indicates the width and height of the picture, and the second argument indicates the co-ordinates of the lower left-hand corner (the origin, if omitted). In both cases, the values are considered to be in terms of the length `\unitlength`. The `picture` environment is in fact just another instance of a box (see **Volume 1** [93, §4.7]) and

10.1 The picture Environment

its contents should only consist of declarations and drawing or positioning commands.

The most commonly used of these commands is:

`\put(<x>,<y>){<object>}`

Definition

which puts $\langle object \rangle$ at the co-ordinates specified by $(\langle x \rangle, \langle y \rangle)$ (which are again in terms of `\unitlength`). The $\langle object \rangle$ may be text, included graphics (using the `graphicx` package's `\includegraphics` command), straight lines or arrows. A straight line is specified by:

`\line(<h>,<v>){<length>}`

Definition

and an arrow (a straight line with an arrowhead) is specified by:

`\vector(<h>,<v>){<length>}`

Definition

There are only a limited number of gradients available. In both cases, the gradient is specified via the horizontal $\langle h \rangle$ and vertical $\langle v \rangle$ displacements, where $\langle h \rangle$ and $\langle v \rangle$ are both integers without a common divisor. In the case of `\line`, $\langle h \rangle$ and $\langle v \rangle$ are restricted to values between -6 and $+6$, inclusive, whereas in the case of `\vector`, those arguments are restricted to values between -4 and $+4$, inclusive.

10.1 The picture Environment

EXAMPLE:

The following code first sets the unit length to 1 cm and then creates a picture that's 3 cm wide by 2.5 cm high. I've added a border around the picture using `\fbox`.

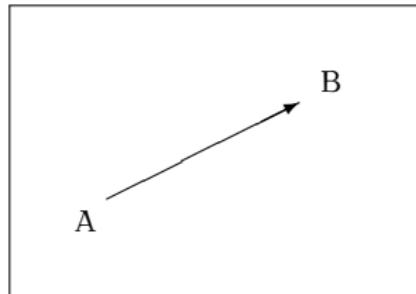
↑ Input

```
\setlength{\unitlength}{1cm}% set the unit length
\fbox{%
  \begin{picture}(3.5,2.5)
    \put(0.5,0.5){A}
    \put(0.8,0.8){\vector(2,1){1.8}}
    \put(2.8,1.8){B}
  \end{picture}
}
```

↓ Input

This produces:

10.1 The picture Environment



Output

Slanted lines are drawn using a special font where the characters consist of small line segments. This is why there's a restriction on the available gradients.

You can also put circles or ovals in the picture using:

`\circle{<diameter>}`

Definition

to create a circle with the given diameter, or

`\oval{<w>}{<h>} [<segment>]`

Definition

to create an oval whose width and height are given by $\langle w \rangle$ and $\langle h \rangle$. In both cases, the lengths are again specified in terms of `\unitlength`. The optional argument of `\oval` may be used if only a quarter or half oval is required, instead of the full oval. In the case of a half oval, $\langle segment \rangle$

10.1 The picture Environment

should be a single letter identifying which half: l (left), r (right), t (top) or b (bottom). For a quarter oval, `<segment>` should be a two-letter combination, for example, `tr` indicates top right.

A filled circle is created using the starred form:

`\circle*`{<diameter>}

Definition

EXAMPLE:

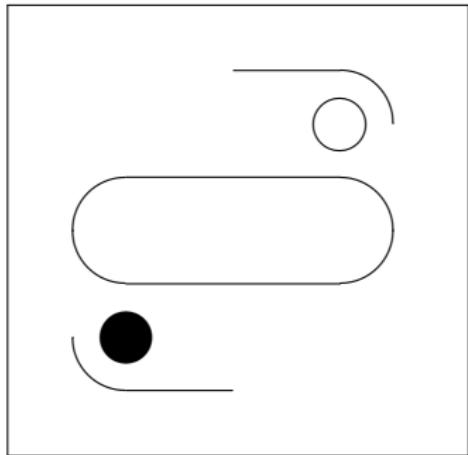
```
\setlength{\unitlength}{1cm}
\fbox{%
\begin{picture}(4.0,4.0)
\put(1,1){\circle*{0.5}}
\put(2,1){\oval(3,1)[bl]}
\put(2,3){\oval(3,1)[tr]}
\put(2,2){\oval(3,1)}
\put(3,3){\circle{0.5}}
\end{picture}
}
```

↑ Input

↓ Input

10.1 The picture Environment

This produces:



Output

`\shortstack[⟨align⟩]{⟨text⟩}`

Definition

The `\shortstack` command is similar to a single-column `tabular` environment, where the contents are given in `⟨text⟩`. As with `tabular`, this command creates a box containing the tabulated data where the rows are separated using `\backslash` but unlike `tabular`, the rows aren't evenly spaced. The optional argument `⟨align⟩` indicates the horizontal alignment of the column. This may be one of `c` (centre), `l` (left) or `r` (right). The default is `c`. The box's refer-

10.1 The picture Environment

ence point is the lower-left corner. When `\shortstack` is used within the `<object>` argument of `\put`, the co-ordinates correspond to the reference point.

EXAMPLE:

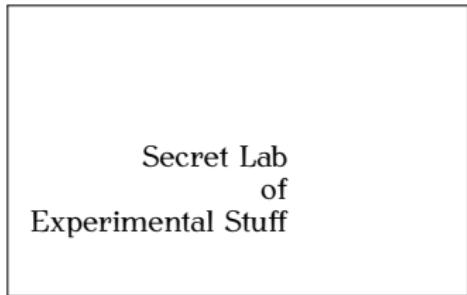
```
\setlength{\unitlength}{1cm}% set the unit length
\fbox{%
\begin{picture}(4,2.5)
\put(0.1,0.5){\shortstack[r]{Secret Lab\\
of\\
Experimental Stuff}}
\end{picture}
}
```

↑ Input

↓ Input

This produces:

10.1 The *picture* Environment



Output

You may remember `\framebox` and `\makebox` from [Volume 1](#) [93, §4.7]. When used within the `picture` environment, these commands have different syntax.

`\framebox(<w>,<h>)[<align>]{<text>}`

Definition

and

`\makebox(<w>,<h>)[<align>]{<text>}`

Definition

In both cases the reference point is the lower-left corner of the box. The width and height of the box are given by `<w>` and `<h>` (again in terms of `\unitlength`). The optional argument `<align>` indicates the alignment of the text. The default is to centre the text both vertically and horizontally. To change this, the `<align>` argument may be one or two letters: `l` (left), `r` (right), `t` (top) and `b` (bottom).

10.1 The picture Environment

There's another box similar to `\framebox`:

```
\dashbox{<dash length>}(<w>,<h>)[<align>]{<text>}
```

Definition

This mostly has the same syntax as above, but produces a dashed frame. The additional argument `<dash length>` specifies the length of the dashes.

EXAMPLE:

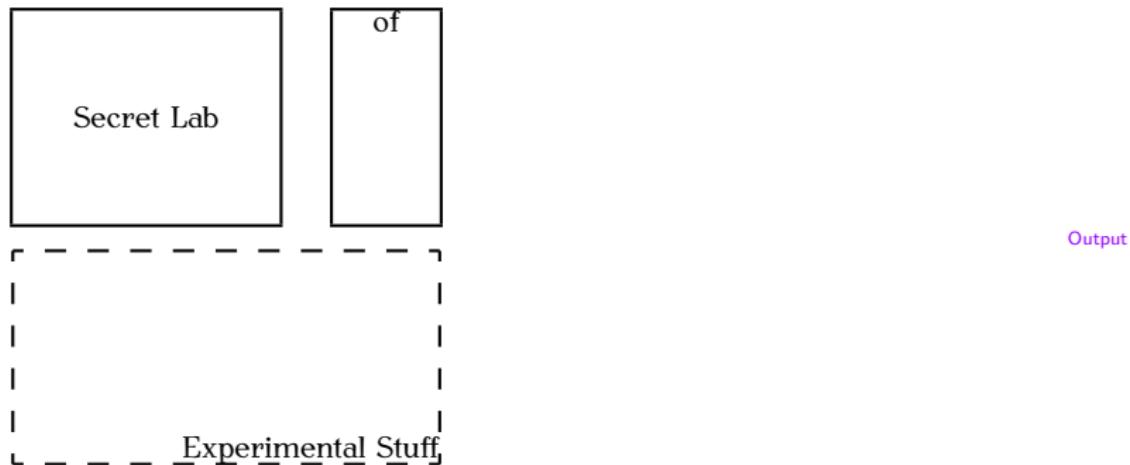
```
\setlength{\unitlength}{1cm}%
\begin{picture}(4.0,4.25)
    \thicklines
    \put(0.0,2.25){\framebox(2.5,2){Secret Lab}}
    \put(3.0,2.25){\framebox(1,2)[t]{of}}
    \put(0.0,0.0){\dashbox{0.2}(3.75,2)[br]{Experimental Stuff}}
\end{picture}
```

↑ Input

↓ Input

This produces:

10.1 The picture Environment



Output

There are two other commands provided for use in the `picture` environment. These aren't used with `\put`. As before, all lengths are in terms of `\unitlength`.

`\qbezier[<points>](<start x>,<start y>)(<control x>,<control y>)(<end x>,<end y>)`

Definition

This draws a quadratic Bézier curve with the given start, end and curvature control points. \LaTeX draws the curve using multiple points. More points results in a smoother curve but a longer document build time. You can

10.1 The picture Environment

use the optional argument to specify the number of points used to draw the curve.

`\multiput(<x>,<y>)(<inc x>,<inc y>){<n>}{<object>}`

Definition

This puts $<n>$ copies of $<\text{object}>$, starting at position $(<x>,<y>)$ and advancing the position by $(<\text{inc } x>,<\text{inc } y>)$ each time.

EXAMPLE 49. A POSTCARD

This example uses the `picture` environment to create a simple postcard advertising an event. The `geometry` package [110] is used to set a non-standard paper size (6 in wide by 4 in high with no margins). I also used the `graphicx` package to include the sample image `chicken.png`.

↑ Input

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

\usepackage[papersize={4in,6in},margin={0in,0in}]{geometry}
\usepackage{graphicx}

\pagestyle{empty}

\setlength{\unitlength}{1in}
```

10.1 The picture Environment

```
\begin{document}
\centering
\begin{picture}(4,6)
\put(0,1.25){\makebox(4,3.75){%
    \includegraphics[height=3.75in]{chicken}}}
\put(0,5){\makebox(4,1){\large\bfseries
    Oh No! The Chickens Have Escaped!}}
\put(0,0.5){\makebox(4,0.75){%
    \shortstack{%
        Written by award-winning author Dickie Duck\\
        Illustrated by internationally renown artist Jos\'e Arara\\
        }%
    }%
}
\put(0,0){\makebox(4,0.5){\large\bfseries
    Book Launch 1st August 2014}}
\end{picture}

\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.) The resulting document is

10.2 The ticket Package

shown in [Figure 10.1](#).

10.2 The ticket Package

The ticket package [23] (version 0.4b, 2010-11-30, at the time of writing) can be used to make labels, visiting cards, pins and flash-cards. The ticket settings can be specified in a ticket definition file (with the extension .tdf). This file can then be specified when you load the ticket package:

`\usepackage[⟨tdf-file⟩,⟨other options⟩]{ticket}`

Input

where ⟨tdf-file⟩ is the filename without the .tdf extension.

In the ticket definition file, you can set up the ticket dimensions. Since this package uses the `picture` environment, described [above](#), you can adjust the unit of measurement by changing the value of `\unitlength` in the definition file. Within this file you can also specify the number and layout of tickets using:

`\ticketNumbers{⟨num cols⟩}{⟨num rows⟩}`

Definition

where ⟨num cols⟩ and ⟨num rows⟩ are the number of tickets in the horizontal and vertical directions. The ticket dimensions are specified using:

10.2 The ticket Package

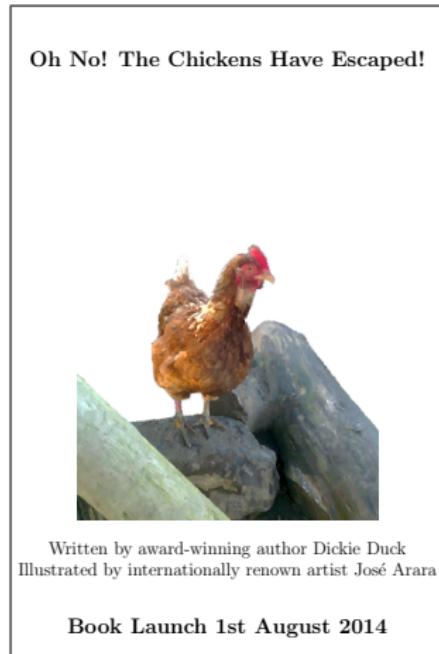


Figure 10.1 A Postcard

10.2 The ticket Package

\ticketSize{<width>}{<height>}

Definition

where the <width> and <height> are in terms of \unitlength. The distance between the tickets is specified using:

\ticketDistance{<x-dist>}{<y-dist>}

Definition

where <x-dist> and <y-dist> are the horizontal and vertical distances in terms of \unitlength. Note that you need to set \unitlength before using the above dimension commands.

For single use only, you can just put these settings in your document after you load the ticket package (without specifying the <tdf-file> in the package options).

In the document you can set up the default ticket content by redefining:

\ticketdefault

Definition

Since this is placed inside the `picture` environment, remember to use picture commands, such as `\put`. The default definition is:

\put (5, 5){Ticket....}

Input

10.2 The *ticket* Package

EXAMPLE

Suppose each ticket should have a logo (stored in the image file `logo.png`) and departmental information:

```
\renewcommand*{\ticketdefault}{%
\put (80,82) {\includegraphics[width=12mm]{logo}}
\put (5,85) {\large\bfseries Secret Lab of Experimental Stuff}
}
```

↑ Input

↓ Input

(Remember to load the `graphicx` package.)

The actual ticket is displayed using

```
\ticket{(content)}
```

Definition

where `<content>` is additional content. You either need to have multiple `\ticket` commands for each ticket or place `\ticket` inside a loop. (Recall §2.7.)

EXAMPLE

Suppose I want to display a ticket with a name on it:

10.2 The *ticket* Package

↑ Input

```
\ticket
{
  \put (49,30) {\makebox(0,0) {\Large\bfseries Polly Parrot}}
}
```

↓ Input

This will create a ticket with the default background (as given by `\ticketdefault`) and the name added to it.

The ticket package documentation suggests defining a wrapper command:

↑ Input

```
\newcommand*{\myticket}[1]{%
  \ticket
  {
    \put (49,30) {\makebox(0,0) {\Large\bfseries #1}}
  }%
```

↓ Input

10.2 The ticket Package

Now you can do, for example,

```
\myticket{Polly Parrot}  
\myticket{Mabel Canary}
```

↑ Input

↓ Input

The ticket package automatically sets the page style to empty. Package options are available to create marks or decorations around the tickets:

crossmark Puts a cross at all four corners.

circlemark Puts an unfilled circle at all four corners.

emptycrossmark Like **crossmark** but only draws the parts of the marker that lie outside the ticket.

cutmark Just adds cutmarks at the outer region.

boxed Adds a frame around the ticket.

10.2 The ticket Package

EXAMPLE 50. NAME LABELS (ticket PACKAGE)

This example creates a set of six name labels. I've used the geometry package to set up the paper margins and the lmodern package [39] to switch to the Latin Modern fonts. The graphicx package is required for the logo. I've used the sample logo `dummy-logo.png`, but you can replace this with another image if you like.

↑ Input

```
\documentclass[a4paper]{article}

% fonts and encodings
\usepackage{lmodern}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[cutmark]{ticket}
\usepackage[margin=5mm]{geometry}
\usepackage{graphicx}

\setlength{\unitlength}{1mm}
\ticketNumbers{2}{3}
\ticketSize{98}{90}
```

10.2 The *ticket* Package

```
\ticketDistance{4}{4}

\renewcommand*\ticketdefault{%
\put (80,80) {\includegraphics[width=12mm]{dummy-logo}}
\put (5,85) {\large\bfseries Secret Lab of Experimental Stuff}
\put (5,75) {\large\scshape University of Somewhere}
\put (45,30) {\makebox(0,0){\Large\itshape Culinary Experimental
Research}}
}

\newcommand*\myticket[1]{%
\ticket
{
\put (45,50) {\makebox(0,0) {\Large\bfseries #1}}
}%
}

\begin{document}
\myticket{Polly Parrot}
\myticket{Mabel Canary}
\myticket{Zöe Zebra}
\myticket{José Arara}
```

10.3 The `leaflet` Class

```
\myticket{Dickie Duck}  
\myticket{Fred Canary}  
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.) The resulting document is shown in [Figure 10.2](#).

EXERCISE 27. NAME LABELS (ITERATION)

Modify the document in [Example 50](#) so that it uses an iteration method to display the tickets. You can use a [comma-separated list](#) with one of the etoolbox commands described in [§2.7.2](#). (You can [download](#) or [view](#) a solution.) Alternatively load the names from the sample [CSV](#) or [SQL](#) data and iterate through the database, as described in [§2.7.1](#). (You can [download](#) or [view](#) a solution for the CSV data.)

10.3 ■ The `leaflet` Class

The `leaflet` class manual [64] is accessed using

10.3 The *leaflet* Class



Figure 10.2 Name Labels (ticket package)

10.3 The *leaflet* Class

```
texdoc leaflet-manual
```

Shell

If you just do

```
texdoc leaflet
```

Shell

you'll get the documented code instead.

The manual is formatted as a z-fold leaflet which illustrates the layout but makes on-screen reading difficult as the reverse sheet is upside-down.

The *leaflet* class loads the *article* class, but some changes are made to the defaults. For example, `\part` is not available and the other sectioning commands aren't numbered. You can use the `letterpaper` or `a4paper` class options to set the paper size, but other paper sizes may need to have the margins adjusted, which can be done using:

```
\setmargins{<top>}{{<bottom>}}{<left>}{{<right>}}
```

Definition

Marginal notes and two-column typesetting are disabled and by default there are no page headers or footers. Paragraph indentation is set to zero and paragraphs are separated by vertical space.

Class options (in addition to the options provided by *article*) are:

`tumble` Print the back sheet upside-down (default).

`notumble` Don't print the back sheet upside-down.

10.3 The *leaflet* Class

- bothsides** Create both the front and back sheet (default).
- frontside** Only create the front sheet.
- backside** Only create the back sheet.
- foldmark** Print a fold mark (default).
- nofoldmark** Don't print a fold mark.
- combine** Combine three pages to a sheet (default). An error is issued if too much text is generated to fit onto the front and back sheets. (The surplus text is ignored.)
- nocombine** Don't combine multiple pages onto a single sheet.
- twopart** Creates a four-page leaflet (first part) and a two-page detachable form (second part).
- notwopart** Not a two-part leaflet (default).

You may find that ragged-right justification produces better results given the narrow page sizes (where a page is one-third of a sheet). The *leaflet* class provides the following commands:

10.3 The *leaflet* Class

\CutLine{\langle page number \rangle}

Definition

This command may only be used in the preamble and indicates that a cut line should be drawn to the left of the page given by *\langle page number \rangle*. The starred version just draws a dotted line. The unstarred version draws a dotted line with a pair of scissors.

\AddToBackground{\langle page number \rangle}{\langle picture code \rangle}

Definition

Again this command may only be used in the preamble. This indicates that *\langle picture code \rangle* should be added to the page given by *\langle page number \rangle*. With the starred version, the *\langle page number \rangle* refers to the sheet number (1 for the front and 2 for the back sheet). The background is placed inside a `picture` environment, so the *\langle picture code \rangle* may include any of the commands described in §10.1. Remember that the co-ordinates are in terms of `\unitlength`. If you want to provide a specific length that's independent of `\unitlength`, the *leaflet* class provides:

\LenToUnit{\langle length \rangle}

Definition

which may be used to specify a length.

The font declaration used for the sectioning commands is given by

\sectfont

Definition

10.3 The *leaflet* Class

This may be redefined as required. The default value is `\bfseries`. The font declaration used for the item label in the `description` environment is given by

`\descfont`

Definition

This may be redefined as required. The default value is `\bfseries`.

EXAMPLE 51. SAMPLE LEAFLET

This example uses the starred version of `\AddToBackground` to place text across the first sheet and uses the unstarred version to place an oval on the first page. The origin is the lower left hand corner of the sheet/page. Rather than working out the co-ordinates in terms of `\unitlength`, I've used `\LenToUnit` with multiples of `\paperwidth` and `\paperheight`.

I've also changed the fonts used by the sectioning commands and the `description` item labels. Since the default Computer Modern fonts don't support bold smallcaps, I've used the Alegreya package [105] to switch to the Alegreya font, which does have bold smallcaps. (For other fonts, have a look at the Font Catalogue [49].) The `wasymp` package [42] provides the `\Square` command used for the tick boxes

Leaflets and flyers tend to be less structured than normal documents, so I've used some commands that typically shouldn't be used (or, at least, used only sparingly on the final copy) in article, report or book-like documents.

10.3 The *leaflet* Class

These include

\newpage

Definition

which forces a page break, without attempting to vertically justify the page,

\bigskip

Definition

which inserts a vertical space,

\vfill

Definition

which inserts a vertical space that will expand to fit the available height. The paragraph justification can be made through an environment, such as `center` or `flushright`, which additionally inserts a vertical space above and below the environment, or the justification can be made through a declaration, such as `\raggedright` (recall [Volume 1 \[93, §2.12\]](#)).

```
\documentclass[a4paper,12pt,notumble]{leaflet}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{Alegreya}
```

↑ Input

10.3 The *leaflet* Class

```
\usepackage{wasy sym}
\usepackage{xcolor}
\usepackage{graphicx}

\renewcommand*\sectfont{\scshape}
\renewcommand*\descfont{\bfseries\scshape}

\AddToBackground{1}{%
\put (0,0)
{\rotatebox{33}{\resizebox{30cm} {!}{\color{lightgray}CLASSIFIED}}}}
}

\AddToBackground{1}{%
\put
(\LenToUnit{0.5\paperwidth}, \LenToUnit{0.5\paperheight})
{\oval(\LenToUnit{0.95\paperwidth}, \LenToUnit{0.95\paperheight})}}
}

\CutLine{6}

\begin{document}
```

10.3 The *leaflet* Class

```
\begin{center}\bfseries\Huge
Culinary Experimental Research

\vfill

\normalsize
\begin{tabular}{c}
Secret Lab of Experimental Stuff\\
University of Somewhere
\end{tabular}

\vfill

\begin{tabular}{c}
Department of Stripy Confectioners\\
College of Somewhere Else
\end{tabular}

\vfill

\includegraphics[height=3cm]{dummy-logo}
\end{center}
```

10.3 The *leaflet* Class

```
\newpage
\raggedright

\section{Secret Lab of Experimental Stuff}
```

The Secret Lab of Experimental Stuff is a top-secret laboratory whose existence is highly classified so don't tell anyone about it or we'll get really cross with you.

The University of Somewhere denies all knowledge of the Secret Lab of Experimental Stuff, except on Open Days where members of the public may visit the facility and ask questions as long as they consent to a memory wipe when they leave. The memory wipe is harmless (well, we haven't really tested it properly, but no one's complained so far) and your memory of the visit will be replaced by a pleasant recollection of spending the day feeding the ducks in the nearby pond.

```
\begin{flushright}
\includegraphics{mallard}
\end{flushright}
```

10.3 The *leaflet* Class

```
\section{Department of Stripy Confectioners}

% lots of text omitted

\newpage
\section{Query Form}
```

If you'd like to know more about the exciting collaboration between the Secret Lab of Experimental Stuff and the Department of Stripy Confectioners please fill in your details below and post this slip to:

```
\bigskip

\begin{tabular}{@{}l}
Miss Ingperson\\
Secret Lab of Experimental Stuff\\
University of Somewhere\\
Some City\\
AB3 4YZ
\end{tabular}
```

10.3 The *leaflet* Class

\bigskip

\Square__I would like to receive quarterly newsletters.

\Square__I agree to having my memory wiped.

\Square__Yes, I'd really like to feed the ducks.

\bigskip

```
\begin{tabular}{@{}lp{4cm}}
Name: & \dotfill \\
Address: & \dotfill\\
& \dotfill \\
& \dotfill \\
& \dotfill \\
Postcode: & \dotfill\\
Country: & \dotfill\\
Telephone: & \dotfill\\
Mobile: & \dotfill\\
Email: & \dotfill
\end{tabular}
```

10.3 The *leaflet* Class

```
\newpage

\section{Research Team}

\begin{description}
\item[Administrator] Mr Big Head

\item[Assistant Administrator] Dr Bor Ing

\item[Project Co-ordinator] Mabel Canary

\item[Senior Scientists] Dickie Duck, Polly Parrot

\item[Research Assistants] Zöe Zebra, José Arara, Fred Canary

\end{description}

\section{Acknowledgements}
```

The Culinary Experimental Research team would like to thank the following:

10.3 The *leaflet* Class

```
\begin{description}
  \item[University of Somewhere] For something or other
  \item[College of Somewhere Else] For providing bread crumbs
  \item[The Ministry of Top Secret Stuff] For supporting the
    project somehow.
\end{description}
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document, and the sample image files [dummy-logo.png](#), [mallard.png](#) and [goose.png](#).)

The resulting document is shown in [Figure 10.3](#) (first sheet) and [Figure 10.4](#) (second sheet). The small marker line to the right of the first section heading on the second sheet is the folding mark. Try this example first without and then with the `twopart` class option.

10.3 The leaflet Class

QUERY FORM

If you'd like to know more about the exciting collaboration between the Secret Lab of Experimental Stuff and the Department of Stripy Confectioners please fill in your details below and post this slip to:

Miss Ingperson
Secret Lab of Experimental Stuff
University of Somewhere
Some City
AB3 4YZ

I would like to receive quarterly newsletters.
 I agree to having my memory wiped.
 Yes, I'd really like to feed the ducks.

Name:
Address:
.....
.....
Postcode:
Country:
Telephone:
Mobile:
Email:

RESEARCH TEAM

ADMINISTRATOR Mr Big Head
ASSISTANT ADMINISTRATOR Dr Bor Ing
PROJECT CO-ORDINATOR Mabel Canary
SENIOR SCIENTISTS Dickie Duck, Polly Parrot
RESEARCH ASSISTANTS Zee Zebra, José Arara, Fred Canary

ACKNOWLEDGEMENTS

The Culinary Experimental Research team would like to thank the following:

UNIVERSITY OF SOMEWHERE For something or other
COLLEGE OF SOMEWHERE ELSE For providing bread crumbs
THE MINISTRY OF TOP SECRET STUFF For supporting the project somehow.

**Culinary
Experimental
Research**

Secret Lab of Experimental Stuff
University of Somewhere

Department of Stripy Confectioners
College of Somewhere Else

**DUMMY
LOGO**

Figure 10.3 A Sample Leaflet (First Sheet)

10.3 The leaflet Class

SECRET LAB OF EXPERIMENTAL STUFF

The Secret Lab of Experimental Stuff is a top-secret laboratory whose existence is highly classified so don't tell anyone about it or we'll get really cross with you.

The University of Somewhere denies all knowledge of the Secret Lab of Experimental Stuff, except on Open Days where members of the public may visit the facility and ask questions as long as they consent to a memory wipe when they leave. The memory wipe is harmless (well, we haven't really tested it properly, but no one's complained so far) and your memory of the visit will be replaced by a pleasant recollection of spending the day feeding the ducks in the nearby pond.



DEPARTMENT OF STRIPY CONFECTIONERS

The Department of Stripy Confectioners is a department within the College of Somewhere Else. The department is internationally known for its cutting-edge research in the field of stripy confectionery, including humbugs and sticks of rock.

In a spirit of co-operation between the University of Somewhere and the College of Somewhere Else, a number of the department's faculty spend half their time feeding ducks in the pond near the University of Somewhere. The Department of Stripy Confectioners has received a grant of £1 million from the Ministry of Top Secret Stuff to purchase the bread crumbs provided by the University of Somewhere.

MIND-CONTROLLING COOKIES

Preliminary tests have shown that the mind-controlling properties of the mind-controlling cookies are somewhat disappointing, but critics have said that they taste very nice and could they have some more bread for the ducks.

INGREDIENTS

- Self-raising flour;
- Butter;
- Chocolate chips;
- Sugar obtained from secret genetically modified beet.

TELEPATHIC CAKES

Preliminary experiments on the telepathic cakes have revealed an unfortunate side-effect. The full results are in the senior scientist's head. Telepathy is required to view them. Please read the full terms and conditions before use.

INGREDIENTS

The ingredients of the telepathic cakes may be obtained telepathically on consumption of the cake.

EXPLODING CHOCOLATES

One of the junior research assistants suffered injuries during the initial development phase but, after a full and detailed investigation, the health and safety department observed that there are also geese present in the duck pond.



Figure 10.4 A Sample Leaflet (Second Sheet)

10.3 The *leaflet* Class

Although the `\AddToBackground` hook adds code to the `picture` environment, if you prefer to use more advanced image drawing code, such as `tikz` [102] or `pstricks` [117], you can do so.

EXAMPLE 52. LEAFLET (WITH tikz)

The preamble code from [Example 51](#) can be modified to use the `tikz` package. The new preamble is as follows:

```
\documentclass[a4paper,12pt,notumble]{leaflet}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{Alegreya}

\usepackage{wasysym}
\usepackage{x11names}{xcolor}
\usepackage{graphicx}
\usepackage{tikz}
\usetikzlibrary{calc}

\renewcommand*\sectfont{\scshape}
```

↑ Input

10.3 The *leaflet* Class

```
\renewcommand*\descfont{\bfseries\scshape}

\AddToBackground{1}{%
\put (0,0)
{\begin{tikzpicture}
\path[fill=Thistle1,draw=Thistle4,double=Orchid1,line width=2pt]
(0,0) rectangle $(\paperwidth, \paperheight)-(4pt,4pt)$;
\end{tikzpicture}
}%
}

\CutLine{6}
```

↓ Input

The remainder of the document is as before. This uses the `calc` `tikz` library, which enables co-ordinate calculations using the `$` syntax. For example

`$(\paperwidth, \paperheight)-(4pt,4pt)$`

Input

indicates the co-ordinate obtained by subtracting the point `(4 pt, 4 pt)` from the point `(\paperwidth,\paperheight)`. The `calc` library must first be loaded using:

10.4 The *pst-barcode* Package

```
\usetikzlibrary{calc}
```

Input

The above example code also loads the `xcolor` package with the `x11names` option to enable the use of the X11 colour names, such as `Thistle1`. For further details about the syntax of the `\path` command, see the `pgf/tikz` user manual [102]. The first sheet is now as shown in Figure 10.5.

For other possible fancy frames or decorations, have a look at the [decoration topic](#). There is also a non-CTAN tikz-based package called `pgfornament` available from <http://altermundus.com/pages/tkz/ornament> however this will require a manual installation since it's not included in the `TEX` distributions (recall [Volume 1](#) [93, §A]).

10.4 ■ The *pst-barcode* Package

The `pst-barcode` package [114] (version 0.12, 2013-10-26, at the time of writing) is a `pstricks` package for drawing twenty-nine different types of bar codes, including EAN-13 and QR codes. Since this is a `pstricks` package, it uses PostScript code, which means that it doesn't work directly with PDF_LATE_X unless you have the `shell escape` enabled and use a package such as `pdfrtricks` [67].

10.4 The *pst-barcode* Package

<p>QUERY FORM</p> <p>If you'd like to know more about the exciting collaboration between the Secret Lab of Experimental Stuff and the Department of Stripy Confectioners please fill in your details below and post this slip to:</p> <p>Miss Ingperson Secret Lab of Experimental Stuff University of Somewhere Some City AB3 4YZ</p> <p><input type="checkbox"/> I would like to receive quarterly newsletters. <input type="checkbox"/> I agree to having my memory wiped. <input type="checkbox"/> Yes, I'd really like to feed the ducks.</p> <p>Name: Address: Postcode: Country: Telephone: Mobile: Email:</p>	<p>RESEARCH TEAM</p> <p>ADMINISTRATOR Mr Big Head ASSISTANT ADMINISTRATOR Dr Bor Ing PROJECT CO-ORDINATOR Mabel Canary SENIOR SCIENTISTS Dickie Duck, Polly Parrot RESEARCH ASSISTANTS Zöe Zebra, José Arara, Fred Canary</p> <p>ACKNOWLEDGEMENTS</p> <p>The Culinary Experimental Research team would like to thank the following:</p> <p>UNIVERSITY OF SOMEWHERE For something or other COLLEGE OF SOMEWHERE ELSE For providing bread crumbs THE MINISTRY OF TOP SECRET STUFF For supporting the project somehow.</p>	<p>Culinary Experimental Research</p> <p>Secret Lab of Experimental Stuff University of Somewhere</p> <p>Department of Stripy Confectioners College of Somewhere Else</p> <p>DUMMY LOGO</p>
--	---	---

Figure 10.5 A Sample Leaflet Using *tikz* (First Sheet)

10.4 The *pst-barcode* Package

There are essentially two options if you want to generate a PDF file and you don't have the `shell escape` enabled:

1. Use `latex`, `dvips` and `ps2pdf` to obtain a PDF version of the document. For example, if your document is in the file `myDoc.tex` then you need to run the following commands:

```
latex myDoc  
dvips -o myDoc.ps myDoc.dvi  
ps2pdf myDoc.ps myDoc.pdf
```

Shell

If you use a frontend, such as TeXworks, you need to find the appropriate buttons or menu options to run these commands. If you use `arara`, you need the following directives:

```
% arara: latex  
% arara: dvips  
% arara: ps2pdf
```

↑ Input

10.4 The *pst-barcode* Package

You can replace the two steps dvips and ps2pdf with a single call to dvipdfm.

2. Put the pstricks code in a standalone document, compile that document using latex, dvips and ps2pdf, as described above, and include the generated PDF file into the main document using \includegraphics.

The *pst-pdf* package [63] can be used to simplify the second option if you have multiple pstricks images in your document, but you still need the latex, dvips and ps2pdf invocations.

If I'm designing a flyer that requires a bar code, such as an advance information sheet with a QR code, I usually use the second option. Once I've generated the bar code, I rarely need to change it, as my modifications to the document usually concern the accompanying text rather than the bar code, so it's easiest to treat the bar code as an external graphics file.

The *pst-barcode* package provides the command:

\psbarcode[*options*]{*text or filename*}{*PS options*}{*type*}

Definition

This generates a bar code with zero size, which means it typically needs to go inside an environment or command where you can specify the height and width. (Recall *Volume 1* [93, §4.7].) Since *pst-barcode* automatically loads the pstricks package [117], you can use the *pspicture* environment:

10.4 The *pst-barcode* Package

```
\begin{pspicture}[(baseline)](llx),(lly)(urx),(ury))

\end{pspicture}
```

Definition

As with the `picture` environment, the co-ordinate arguments are specified using parentheses, but take care as the syntax of the `pspicture` environment is slightly different to that of the `picture` environment. In the case of `pspicture`, the first argument in parentheses (llx, lly) specifies the co-ordinates for the lower left corner, and second argument in parentheses (urx, ury) specifies the co-ordinates for the upper right corner of the picture's bounding box. If (llx, lly) is omitted, the origin is assumed.

The optional argument $\langle \text{options} \rangle$ of `\psbarcode` is a `key=value` list. Available options include:

- file** This is a `boolean key`. This determines whether the argument $\langle \text{text or filename} \rangle$ is the bar code text (in the case of `file=false`) or the name of the file containing the bar code text (in the case of `file=true`). The default value for this option is `file=false`.
- transx** This specifies a horizontal shift to apply to the bar code. The default value is `transx=0`.
- transy** This specifies a vertical shift to apply to the bar code. The default value is `transy=0`.

10.4 The *pst-barcode* Package

- scalex** This specifies a horizontal scaling to apply to the bar code. The default value is `scalex=1`.
- scaley** This specifies a vertical scaling to apply to the bar code. The default value is `scaley=1`.
- rotate** This specifies the rotation (in degrees) to apply to the bar code. The default value is `rotate=0`.

The *<PS options>* argument are PostScript options separated by [white-space](#). Available options include:

- includetext** This enables human readable text.
- font** This sets the font, which must be a PostScript font. The default is `/Helvetica`.
- guardwhitespace** This enables the display of whitespace guard marks.

The final argument *<type>* of [`\psbarcode`](#) indicates the type of bar code. For example, `ean13` for an EAN-13 bar code, `isbn` for an ISBN bar code, or `qrcode` for a QR code.

10.4 The *pst-barcode* Package

EXAMPLE 53. ISBN BAR CODE

The ISBN bar code is just a special form of EAN-13 bar code with a particular prefix. The data, provided in the `(text or filename)` argument, should contain 9 or 10 digits for ISBN-10, and 12 or 13 digits for ISBN-13. (In both cases, the digits separated appropriately with hyphens.) If only 9 (ISBN-10) or 12 (ISBN-13) digits are specified the ISBN check digit is calculated automatically.

The ISBN for the paperback version of this book is 978-1-909440-07-4 so I can create the ISBN bar code using:

```
\psbarcode{1-909440-07-4}{includetext guardwhitespace}{isbn}
```

Input

Here's a complete document containing the bar code:

```
\documentclass{article}

\usepackage{pst-barcode}

\begin{document}

\begin{pspicture}(-.4,-.2)(3.8,3)
\psbarcode{1-909440-07-4}{includetext guardwhitespace}{isbn}

```

↑ Input

10.4 The *pst-barcode* Package

```
\end{pspicture}
```

```
\end{document}
```

↓ Input

How did I work out the co-ordinates for the bounding box? I put the picture inside the argument of `\frame`, which marks the picture's extent with a rectangle and then adjusted the co-ordinates until the picture fitted inside the frame. Like this:

```
\frame{%
\begin{pspicture}(-.4,-.2)(3.8,3)
\psbarcode{1-909440-07-4}{includeText guardwhitespace}{isbn}
\end{pspicture}%
}
```

↑ Input

↓ Input

Remember that this example document must be compiled with `latex` rather than `pdflatex`. If you want to turn this into an image that you can include in another document, change the document class to `standalone` [82] and remove all unnecessary blank lines:

10.4 The *pst-barcode* Package

↑ Input

```
% arara: latex
% arara: dvips
% arara: ps2pdf
\documentclass{standalone}
\usepackage{pst-barcode}
\begin{document}
\begin{pspicture}(-.4,-.2)(3.8,3)
\psbarcode{1-909440-07-4}{includeText guardwhitespace}{isbn}
\end{pspicture}
\end{document}
```

↓ Input

(You can [download](#) or [view](#) this document.)

If the file is called `barcode-isbn.tex` then you need to run:

Shell

```
latex barcode-isbn
dvips -o barcode-isbn.ps barcode-isbn.dvi
ps2pdf barcode-isbn.ps barcode-isbn.pdf
```

Alternatively, if you use `arara` and have included the `arara` directives shown above, you can just do:

10.4 The *pst-barcode* Package

```
arara barcode-isbn
```

Shell

This creates a PDF file called `barcode-isbn.pdf` that you can now include in another document using:

```
\includegraphics{barcode-isbn}
```

Input

This produces:

ISBN 978-1-909440-07-4



Output

EXERCISE 28. A QR Code

A QR code is obtained by setting `<type>` to `qrcode`. The data setting `<text or filename>` is typically a website address, so for this exercise, adapt the code from [Example 53](#) so that it's now a QR code. Set the data to a website of your choice. If you can't think of one, you can use this book's home page: <http://www.dickimaw-books.com/latex/admin/>.

You can [download](#) or [view](#) a solution to this exercise.

10.5 ■ The `flowfram` Package and the `flowframtk` Application

The `LATEX` kernel provides a single-column mode through the use of `\onecolumn` (or the `onecolumn` class option) and two-column mode through the use of `\twocolumn` (or the `twocolumn` class option). The `flowfram` package provides a way to extend this so that you can have an arbitrary number of columns of arbitrary width and height placed in arbitrary locations.

The standard `LATEX` declarations `\onecolumn` and `\twocolumn` automatically insert a page break before switching modes. One of the reasons for this page break is to ensure that there isn't a paragraph spanning different width columns as `TEX`'s output routine doesn't set the line width until the

10.5 The *flowfram* Package and the *flowframtk* Application

end of the paragraph which would leave the tail end of the paragraph with the incorrect width at the start of the new column. The *flowfram* package doesn't automatically insert page breaks in this manner, but this inherent problem caused by the asynchronous behaviour of TeX's output routine is present if you have a paragraph spanning different width columns that have been defined using *flowfram*, so you need to take care.

These arbitrary columns are termed "flow frames" in the *flowfram* user manual [98], which is accessed using

```
texdoc ffuserguide
```

Shell

(If you just do

```
texdoc flowfram
```

Shell

you'll get the documented code instead.)

There are two other types of frames: "static frames" and "dynamic frames". These two types of frames need to have their contents set explicitly. With static frames, the contents are typeset in a box on being set. With dynamic frames, the contents are stored in a macro and retypeset each time the frame is drawn on a page. The frames are drawn onto the page in the following order: static, flow, dynamic. Within each category, the frames are drawn in order of definition. If frames overlap, their

10.5 The *flowfram* Package and the *flowframtk* Application

contents will overlap. (In other words, the text inside one frame doesn't attempt to avoid collision with the text inside another frame.)

Frames may have a border drawn around them. The default border is just a rectangle, but other borders may be used. The paragraph shape is unaffected by the shape of the border but it can be changed either using TeX's `\parshape primitive` or one of the commands provided by the `shapepar` package [4].

Flow frames can be defined using

`\newflowframe[⟨page list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]`

Definition

Static frames can be defined using

`\newstaticframe[⟨page list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]`

Definition

Dynamic frames can be defined using

`\newdynamicframe[⟨page list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]`

Definition

Each of these commands has a starred version that adds a rectangular border to the frame. The arguments are as follows:

⟨page list⟩ The list of pages on which this frame is visible. This may be one of the keywords: all, none, odd or even; or this may be a `comma-separated list` of page numbers or page ranges. Page

10.5 The *flowfram* Package and the *flowframtk* Application

ranges may be open-ended using $\langle n \rangle$ for pages less than page $\langle n \rangle$ or $\rangle\langle n \rangle$ for pages greater than page $\langle n \rangle$; or the ranges may be closed using $\langle n \rangle-\langle m \rangle$ for pages between $\langle n \rangle$ and $\langle m \rangle$, inclusive. The page numbers referenced in the $\langle page\ list \rangle$ by default refer to the decimal value of the page counter. (For example, 1 means page 1 or page i or page I.) If the `pages=absolute` package option is used, then the page number refers to the absolute page number. If $\langle page\ list \rangle$ is omitted, all is assumed.

- $\langle width \rangle$ The width of the frame.
- $\langle height \rangle$ The height of the frame.
- $\langle x \rangle$ The x -coordinate of the bottom left-hand corner of the frame relative to the typeblock.
- $\langle y \rangle$ The y -coordinate of the bottom left-hand corner of the frame relative to the typeblock.
- $\langle label \rangle$ A label that uniquely identifies this frame.

The typeblock is the area where the text would be typeset if the document was in the regular one-column mode. Each frame can be referenced either by its label or by its index (starting from 1 for each frame type). For

10.5 The *flowfram* Package and the *flowframtk* Application

example, the first flow frame to be defined has an index equal to 1, and the first static frame to be defined also has an index equal to 1.

The contents of a static frame can be set using:

```
\setstaticcontents{\langle id\rangle}{\langle contents\rangle}
```

Definition

where the frame contents are given by $\langle \text{contents} \rangle$. For the unstarred version, $\langle \text{id} \rangle$ is the frame's index. For the starred version, $\langle \text{id} \rangle$ is the label. Alternatively, the contents can be set using the `staticcontents` environment:

```
\begin{staticcontents}{\langle id\rangle}  
  \langle contents\rangle  
\end{staticcontents}
```

Definition

As before there is a starred version where $\langle \text{id} \rangle$ is the frame's label rather than its index.

The contents of a dynamic frame can be set in a similar manner using:

```
\setdynamiccontents{\langle id\rangle}{\langle contents\rangle}
```

Definition

or using the `dynamiccontents` environment:

```
\begin{dynamiccontents}{\langle id\rangle}  
  \langle contents\rangle  
\end{dynamiccontents}
```

Definition

10.5 The *flowfram* Package and the *flowframtk* Application

Unlike the static frames, you can also append text to a dynamic frame using:

`\appenddynamiccontents{<id>}{<text>}`

Definition

As before, these all have starred versions where $\langle id \rangle$ is the frame's label.

 Note that verbatim text isn't permitted in a dynamic frame, even with the environment version. Make sure that you set the contents before the frame is displayed on the page.

EXAMPLE

Static frames are useful for background images, as shown below. This is a simple document that uses the `lipsum` package [33] to generate dummy text. This is a contrived example that shows what happens when you have overlapping frames and what happens if you have a paragraph spanning flow frames of different widths. The document text is placed in the flow frames in the order of the frame definition, which is why the text starts on the right hand frame, as that was the first flow frame to be defined.

```
\documentclass[a4paper]{article}  
  
\usepackage{lipsum}
```

↑ Input

10.5 The *flowfram* Package and the *flowframtk* Application

```
\usepackage{graphicx}
\usepackage{flowfram}

\newflowframe
{0.6\textwidth}%
{0.3\textheight}%
{0.4\textwidth}%
{0.7\textheight}%

\newflowframe{0.6\textwidth}{0.5\textheight}{0pt}
{0.5\textheight}

\newflowframe*{\textwidth}{0.4\textheight}{0pt}{0pt}

\newstaticframe{2in}{2in}{0pt}{0pt}

\setstaticcontents{1}{\includegraphics[height=2in]{chicken} }

\begin{document}

\lipsum[1-4]
```

10.5 The *flowfram* Package and the *flowframtk* Application

```
\end{document}
```

↓ Input

The rather unpleasant result is shown in [Figure 10.6](#).

The issue caused by TeX's asynchronous output routine being unable to adjust the line width over a frame break can be seen by the shortened lines in the end part of the paragraph at the beginning of the lower frame. (The frame with the border.) The *flowfram* package notices the problem and issues a warning with a recommendation:

```
Package flowfram Warning: Moving to flow frame of unequal width,  
(flowfram)           use of \framebreak advised, or text  
might not appear  
correctly (difference = 137.9979pt, tolerance = 2.0pt)
```

As with all manual interventions, the use of

```
\framebreak
```

Definition

before the first word of the new frame (between "magna." and "Nunc" in this example) should only be resorted to on the final version of the document, once all the text has been written.

10.5 The *flowfram* Package and the *flowframtk* Application



Figure 10.6 Overlapping Frames (*flowfram* package)

EXAMPLE 54. ADVANCE INFORMATION SHEET

This example uses the *flowfram* package to create an advance information sheet for a book. The *geometry* package is used to set the margins. The page numbering is suppressed using the empty page style and the section numbering is suppressed by setting the *secnumdepth* counter to 0. (This just saves me from remembering to use the starred version of `\section`.) I've used the *drm* package [29] to illustrate a different font from those I've previously used, and I've used the *pifont*, which provides the *dinglist* environment that was mentioned in Volume 1 [93, §8.2]. Recall `\dimexpr` from §2.1.3, which is used here to calculate the position of the static frame.

↑ Input

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

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{drm}

\usepackage{pifont}
\usepackage[margin=0.5in]{geometry}
\usepackage{graphicx}
\usepackage{flowfram}
```

10.5 The *flowfram* Package and the *flowframtk* Application

```
\pagestyle{empty}
\setcounter{secnumdepth}{0}

\newflowframe{0.6\textwidth}{\textheight}{0pt}{0pt}

\newdynamicframe{.3\textwidth}{\textheight}{.7\textwidth}{0pt}[sidepane]

\newstaticframe{2in}{2in}{\dimexpr(\textwidth-2in)}{0pt}[logo]

\setstaticcontents*{logo}{\includegraphics[width=2in]{dummy-logo}}

\begin{dynamiccontents*}[sidepane]
{\raggedright\bfseries\scshape\Large
Oh No! The Chickens Have Escaped
\par
}

\centering
\bigskip

\includegraphics[width=\linewidth]{chicken}
```

10.5 The *flowfram* Package and the *flowframtk* Application

```
\bigskip

\begin{tabular}{@{}l}
Genre: & Children's Illustrated \\
        & Fiction\\
RRP:   & £5.99\\
Format: & Paperback\\
Pages:  & 30\\
Pub Date: & 1st August 2014\\
ISBN:   & 978-x-xxxxxx-xx-x
\end{tabular}
```

```
\vfill
```

```
\includegraphics{barcode-qr}
```

```
\vfill
```

```
\end{dynamiccontents*}
```

```
\begin{document}\raggedright
```

10.5 The *flowfram* Package and the *flowframtk* Application

\section{About the Book}

A fun illustrated children's story about some escaped chickens. Fred and Mabel are looking after Granny's chickens for the day but, oh no, they've escaped. Will Fred and Mabel find them all before the chickens get into the road or get eaten by the hungry fox?

\section{About the Author}

Dickie Duck lives somewhere or other and won the best fowl book award in 2014. He likes writing silly stories about ducks and chickens.

\section{Keypoints}

\begin{dinglist}{118}

\item A fun way of teaching children to count.

\item Children will enjoy the repetition and rhyme.

\item Features chickens doing stupid things.

10.5 The *flowfram* Package and the *flowframtk* Application

```
\item Completely fictitious book encourages children's
imagination.
\end{dinglist}

\section{Marketing}

\begin{dinglist}{118}
\item Written by award-winning author.

\item Illustrated by world famous artist.

\item Some other really interesting marketing information.
\end{dinglist}

\section{Contact}

\begin{tabular}{@{}l}
Dickie Duck\\
1 The Street\\
Another Village\\
Some City\\

```

10.5 The *flowfram* Package and the *flowframtk* Application

```
Imagineshire\\
YZ1 2AB
\end{tabular}

\end{document}
```

↓ Input

You can [download](#) or [view](#) this example document. It uses the sample images `chicken.png` and `dummy-logo.png`. It also uses the `barcode-qr.pdf` file created in [Exercise 28](#).

The resulting document is shown in [Figure 10.7](#).

If you find it a bit awkward to work out the dimensions and locations of the frames, there's a helper GUI application called `flowframtk`, which provides a graphical means of defining the frames. As with `datatooltk` and `arara`, this is a Java application so, if you want to use it, make sure you have an up-to-date version of the Java runtime environment installed on your computer.

To install `flowframtk` download the installer from [flowframtk's home page](#). This is a `.jar` file. If your operating system knows how to run a `.jar` file, you should just be able to double-click on it, otherwise you can run it from the command line using:

10.5 The *flowfram* Package and the *flowframtk* Application

About the Book

A fun illustrated children's story about a sunburned chicken, Fred and Mabel who are looking after Grampy's chickens for the day but do not do a very good job. Will Fred and Mabel find them all before the chickens get into the road or get eaten by the hungry fox?

About the Author

Dickie Duck lives somewhere or other and won the best first book award in 2018. He likes writing silly rhymes about ducks and chickens.

Keypoints

- ♦ A fun way of teaching children to count
- ♦ Children will enjoy the repetition and rhyme
- ♦ Features chickens doing stupid things.
- ♦ Completely fictitious book encourages children's imagination.

Marketing

- ♦ Written by award-winning author.
- ♦ Illustrated by world famous artist.
- ♦ Some other really interesting marketing information.

Contact

Dickie Duck
1 The Street
Ardmore Village
Some City
Imaginative
YZ 1AB

OH NO! THE
CHICKENS HAVE
ESCAPED



Genre: Children's Illustrated Fiction
RRP: £5.99
Format: Paperback
Pages: 30
Pub Date: 1st August 2019
ISBN: 978-1-23456-789-0



DUMMY
LOGO

Figure 10.7 Advance Information Sheet

10.5 The *flowfram* Package and the *flowframtk* Application

```
java -jar flowframtk-0.7-installer.jar
```

Shell

(You may need to specify the full path to the .jar file. The version number 0.7 may also need to be changed if a new version has been produced since the time of writing this.)

Once **flowframtk** has been installed, it can be run either from your operating system's applications menu or from the command line using:

```
flowframtk
```

Shell

The main window is shown in [Figure 10.8](#).

EXAMPLE 55. ADVANCE INFORMATION SHEET (WITH FLOWFRAMTK)

To illustrate the use of **flowframtk**, the rest of this section will use **flowframtk** to create the advance information sheet from [Example 54](#). The result will be slightly different as I'm going to add some extra features such as coloured backgrounds.

For this example, I prefer to work in metric measurements so, to reduce rounding errors, I'm first going to set the storage unit to millimetres and the grid to centimetres. To set the storage unit, go to **Settings**→**Configure Image Settings**. This opens the dialog box shown in [Figure 10.9](#). Use the **Storage Unit** drop-down menu to change the unit to mm. Click on the green tick button to save this change and close the dialog window.

10.5 The *flowfram* Package and the *flowframtk* Application

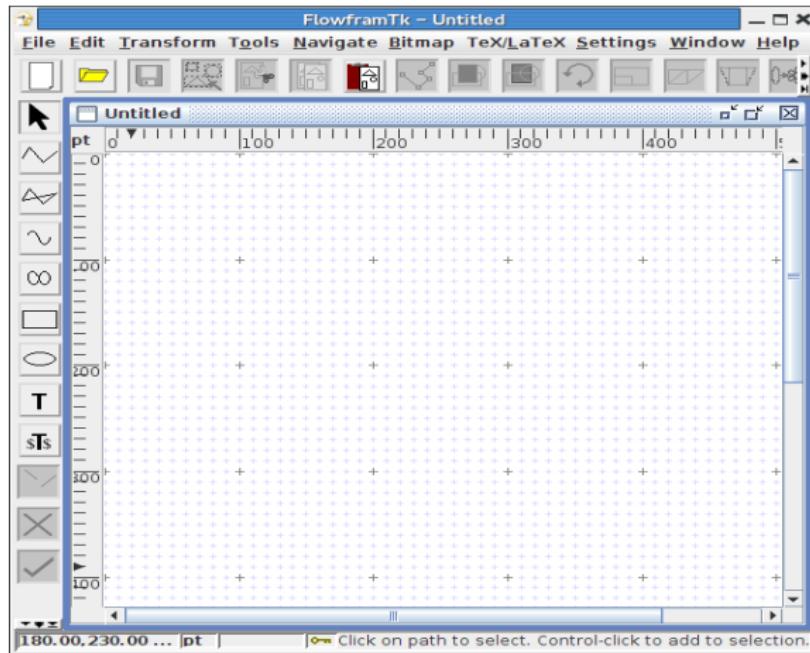


Figure 10.8 FlowframTk Main Window

10.5 The *flowfram* Package and the *flowframtk* Application

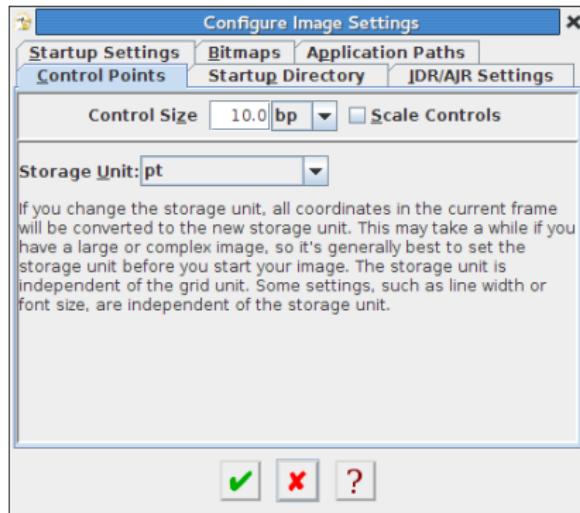


Figure 10.9 FlowframTk — Set the Storage Unit

10.5 The *flowfram* Package and the *flowframtk* Application

Next go to **Settings**→**Grid**→**Grid Settings** which will open the dialog window shown in [Figure 10.10](#). Select the **Rectangular** tab (if not already selected) and change the major divisions to 1 cm and the sub-divisions to 2 or 4. Click on the green tick button to save this change and close the dialog window.

Next make sure the paper size is correctly set. I'm using A4 portrait paper, which can be selected through the **Settings**→**Paper** submenu. Now the **T_EX** settings for the document need to be specified, so go to **Settings**→**Configure TeX/LaTeX Settings** which will open the dialog window shown in [Figure 10.11](#).

Recall from [Example 54](#) that I used the 12pt class option. I can specify that I want this by changing the **Normal Font Size** drop-down menu to 12. I'm going to use the default article class so I've left the **Use default class** radio button selected. If I want to use a different class, for example scrartcl, then I would have to select the **Use class** radio button and type in the class name (without the extension) in the neighbouring field. Again, click on the green tick button to save the changes and close the window.

Next I need to specify the packages I want to use. This is done in the preamble editor, which is opened using the **TeX/LaTeX**→**Preamble** menu item. Note that I don't include the **geometry** or **flowfram** packages since these will automatically be added when I later use the export function. Here are the packages (as from the previous example) and I've also added the code to

10.5 The flowfram Package and the flowframtk Application

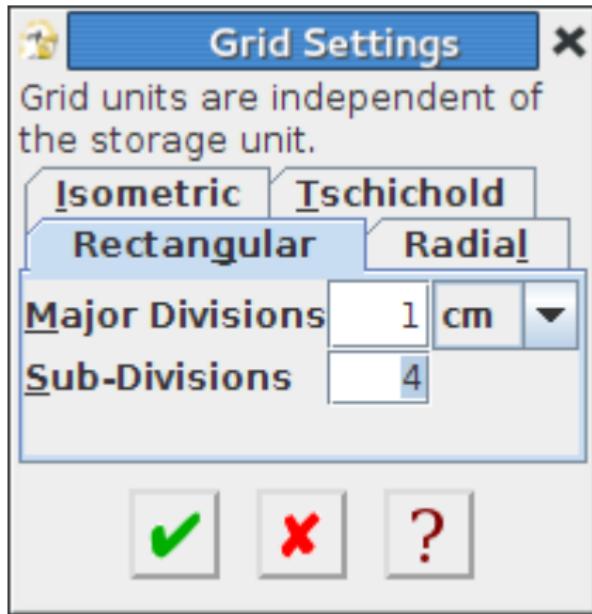


Figure 10.10 FlowframTk — Set the Grid

10.5 The *flowfram* Package and the *flowframtk* Application

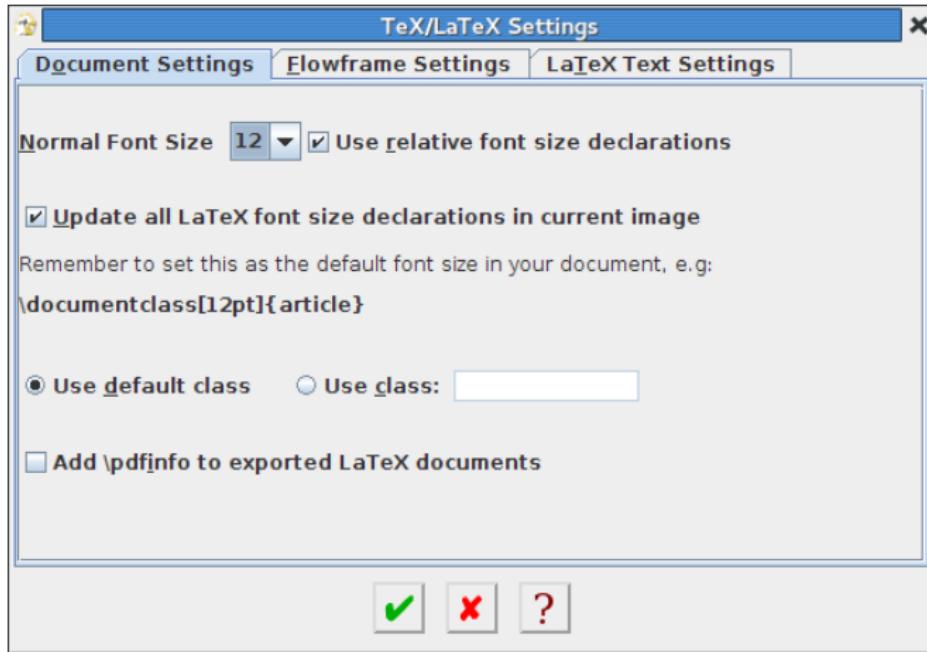


Figure 10.11 FlowframTk — Set the TeX/LaTeX Settings

10.5 The *flowfram* Package and the *flowframtk* Application

set the empty page style and switch off the section numbering, but I've deferred it using `\AtBeginDocument`:

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{drm}

\usepackage{pifont}
\usepackage{graphicx}

\AtBeginDocument{%
    \pagestyle{empty}%
    \setcounter{secnumdepth}{0}%
}
```

↑ Input

See [Figure 10.12](#).

You can choose whether to show or hide the grid using the **Settings**→**Grid** sub-menu. I'm going to hide the grid but keep the rulers visible. Now I need to specify the margins. In [Example 54](#) I had 0.5in margins, but now I want to have a borderless document, so I'm going to set all the margins

↓ Input

10.5 The *flowfram* Package and the *flowframtk* Application

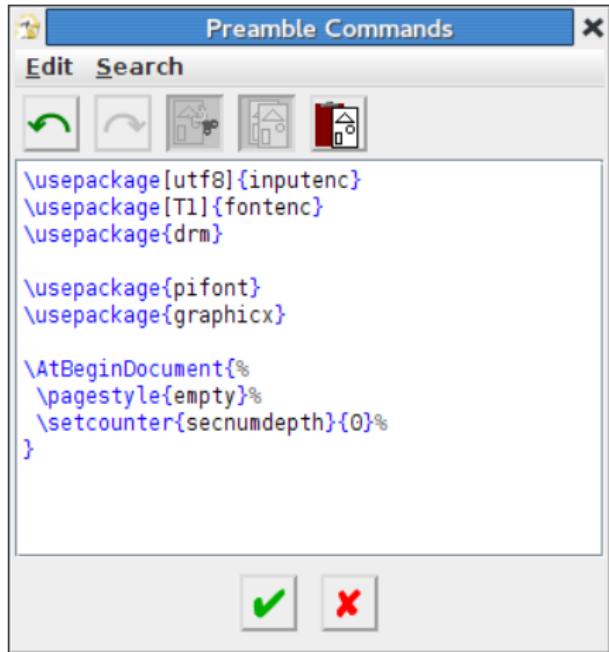


Figure 10.12 FlowframTk — Setting the Preamble

10.5 The *flowfram* Package and the *flowframtk* Application

to zero. This is done using the TeX/LaTeX→Flow Frames→Set Typeblock menu item, which opens the dialog shown in [Figure 10.13](#). Make sure all the margins are set to 0. (For this example, you don't need the other settings in this dialog window.) Although the margins default to 0, you must still click on the green tick button to save and close the dialog. A grey rectangle should now be displayed on the canvas with the word “typeblock” in the bottom left hand corner of the rectangle. (With zero margins, it may be difficult to see the rectangle.)

Now select the rectangle tool either using the Tools→Rectangle menu item or click on the rectangle button in the left tool bar. To draw a rectangle on the canvas, click where you want one corner and move the mouse to the location where the opposite corner should be and click there. For example, in [Figure 10.14](#), I've created a rectangle with top left corner at (0 mm, 0 mm) and bottom right corner at (130 mm, 297 mm).

This rectangle will represent the main flow frame in my document, but first I want to add a fill colour and remove the outline. To do this, I need to switch to the select tool (either using the Tools→Select menu item or by clicking on the button in the side bar with an arrow on it). Then I can click anywhere inside the rectangle to select it. When it's selected, a dashed red rectangle will appear around it. Then I can use the Edit→Fill Colour menu item to open the fill colour selector. In [Figure 10.15](#), I have selected the Colour radio button and specified 20% cyan, 20% magenta, 0% yellow and

10.5 The *flowfram* Package and the *flowframtk* Application

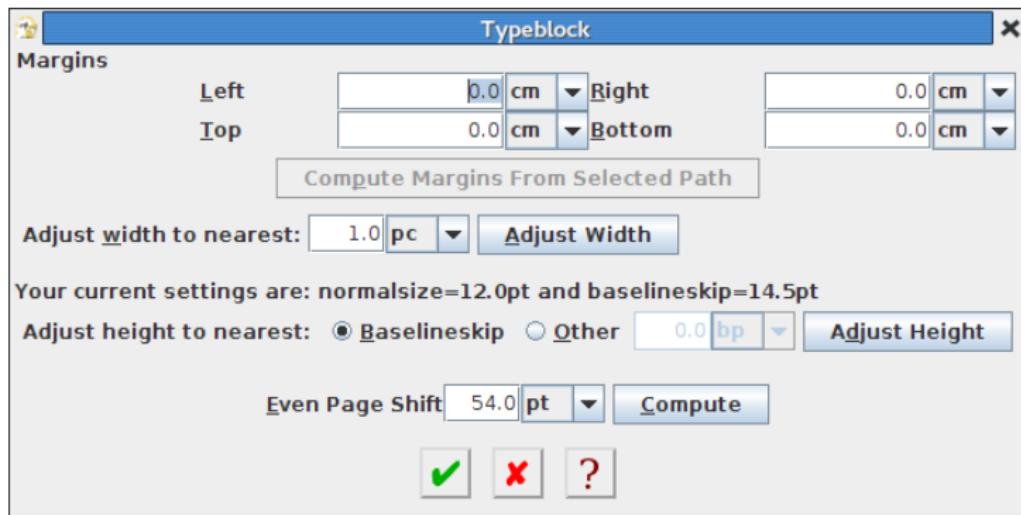


Figure 10.13 FlowframTk — Setting the Margins

10.5 The *flowfram* Package and the *flowframtk* Application

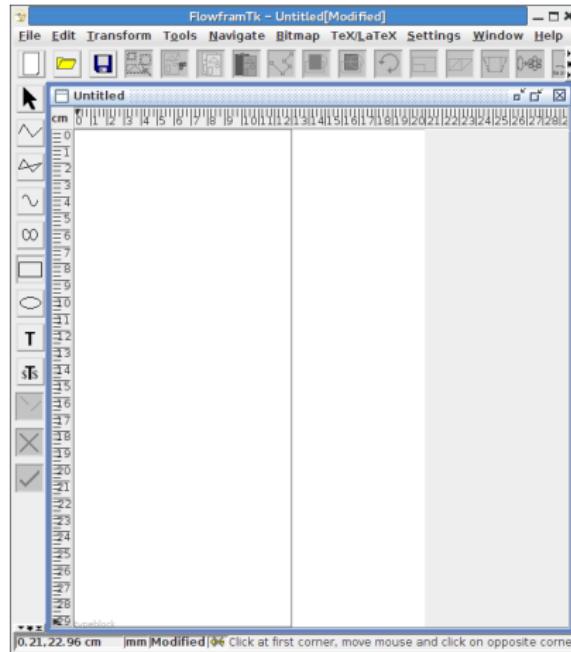


Figure 10.14 FlowframTk — Create a Rectangle

10.5 The *flowfram* Package and the *flowframtk* Application

0% black. The rectangle's black outline can be removed by selecting the Edit→Path→Line Colour menu item, which will open the line colour selector. Click on the Transparent radio button to remove the outline.

This rectangle now needs to be identified as a flow frame. Make sure it's still selected and use the menu item TeX/LaTeX→Flow Frames→Set Frame to open the flow frame selector. Set the Type to Flow and this will enable the flow frame related options. Give the frame a label (for example, "main") and select the Border As Shown option. The margins can also be set to prevent the document text running against the border. I've chosen 5mm for each margin, as shown in [Figure 10.16](#). The even page options can be ignored since the result will be a single-paged document. Again click on the green tick to save the changes and close the dialog window. The selected rectangle in the main window should now have a pale grey rectangle inside it that shows the frame's margins (see [Figure 10.17](#)).

Now I want to create a static frame in the currently unfilled narrow region on the right hand side of the page. This will just provide a background colour. I will later make some other frames on top of this one that will contain text and images. The process is much the same as for the previous frame. The rectangle tool is selected, a rectangle is drawn with opposing corners at (132 mm, 0 mm) and (210 mm, 297 mm). The fill colour is set to 0% cyan, 0% magenta, 50% yellow, 0% black, and the outline is set to transparent. However, the frame should now be a static frame, so

10.5 The *flowfram* Package and the *flowframtk* Application

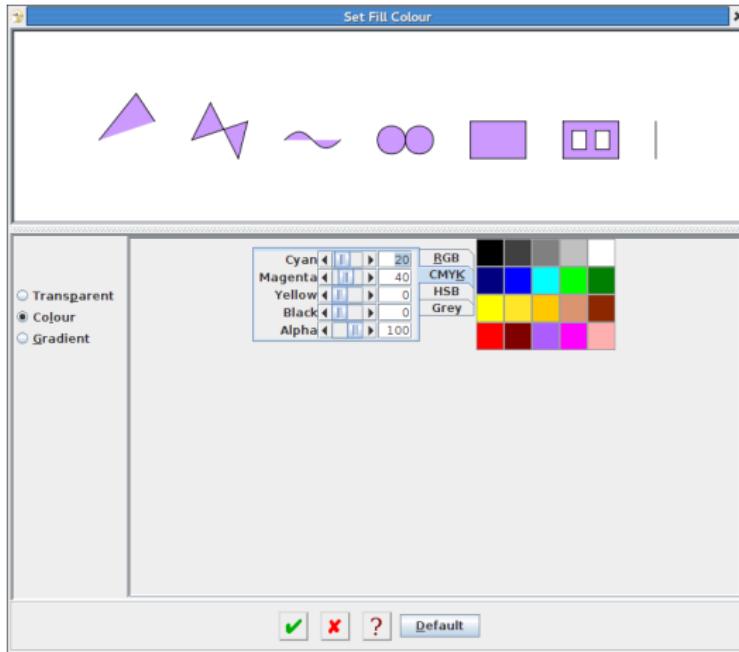


Figure 10.15 FlowframTk — Setting the Fill Colour

10.5 The *flowfram* Package and the *flowframtk* Application

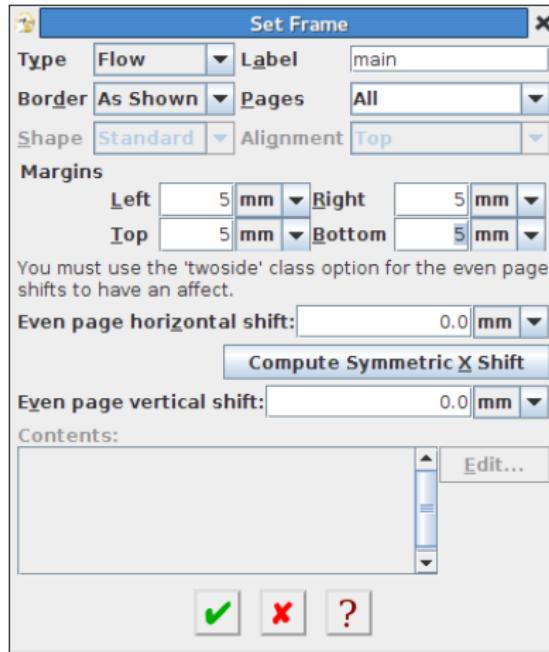


Figure 10.16 FlowframTk — Setting Flow Frame Data

10.5 The *flowfram* Package and the *flowframtk* Application

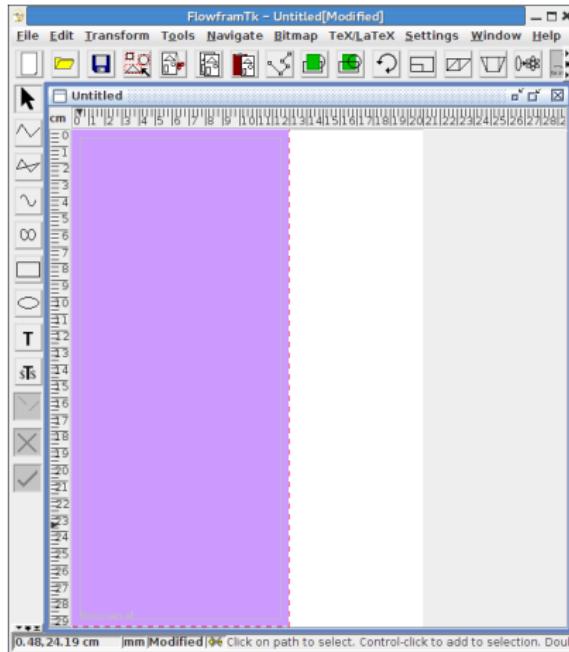


Figure 10.17 FlowframTk — Flow Frame Data Assigned

10.5 The *flowfram* Package and the *flowframtk* Application

once the rectangle has been created, select the menu item $\text{\TeX}/\text{\LaTeX}\rightarrow\text{Flow Frames}\rightarrow\text{Set Frame}$ and set the Type to Static. This enables a different set of options to earlier, but for this frame the only extra information needed is the label. I've set this to "sidepane", as shown in [Figure 10.18](#). Again, click on the green tick button to save the changes and close the dialog window.

It's a good idea at this point to save the image in case something goes wrong. There are two native file formats: `.jdr` (binary) and `.ajr` (ASCII). The binary version has greater precision but the ASCII version works better with version control systems (see [§13.2](#)). Since I use version control and I don't need double-precision for my co-ordinates, I'm going to use the ASCII version (`.ajr`). To save to a new file use the $\text{File}\rightarrow\text{Save As}$ menu item and select the appropriate file type (in my case, *flowframtk* ascii file (`*.ajr`)), as shown in [Figure 10.19](#).

Now I need an area in which to put the book title (which appears on the top right of [Figure 10.7](#)). Again I need to use the rectangle tool to use as a guide for my frame. My rectangle has opposing corners at (136 mm, 5 mm) and (206 mm, 62 mm). In this case I don't need to change the colours as I'm only using the rectangle as a guide to define a borderless dynamic frame. As before, I need to select this new rectangle and use the $\text{\TeX}/\text{\LaTeX}\rightarrow\text{Flow Frames}\rightarrow\text{Set Frame}$ menu item to open the dialog box. Now I set the Type to Dynamic and the label to "title", but this time the Border option needs to be set to None as shown in [Figure 10.20](#).

10.5 The *flowfram* Package and the *flowframtk* Application

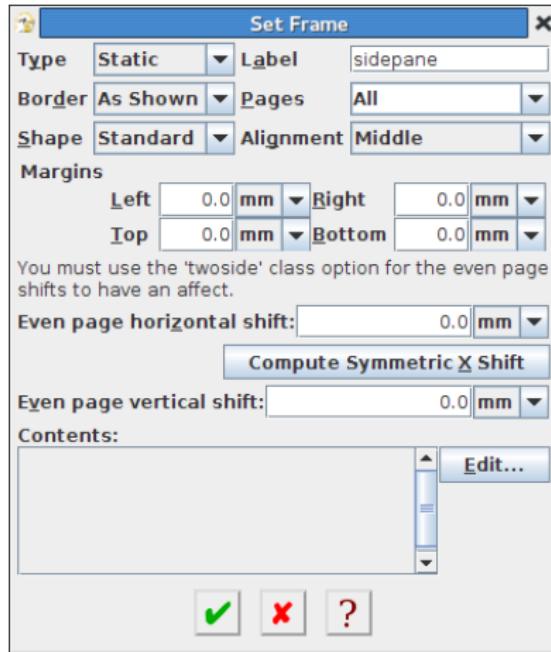


Figure 10.18 FlowframTk — Setting Static Frame Data

10.5 The *flowfram* Package and the *flowframtk* Application

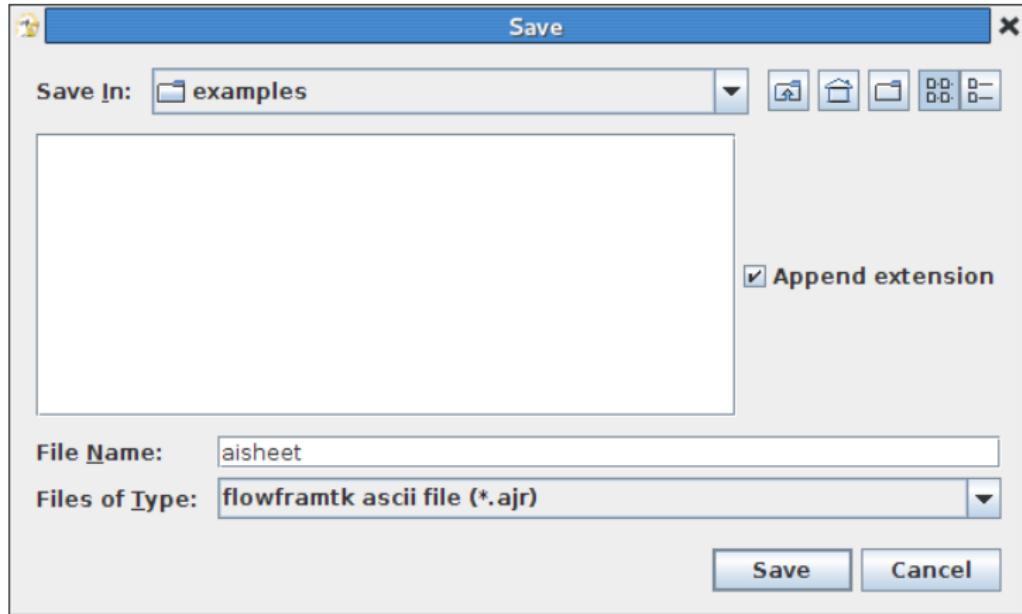


Figure 10.19 FlowframTk — Saving the Image

10.5 The *flowfram* Package and the *flowframtk* Application

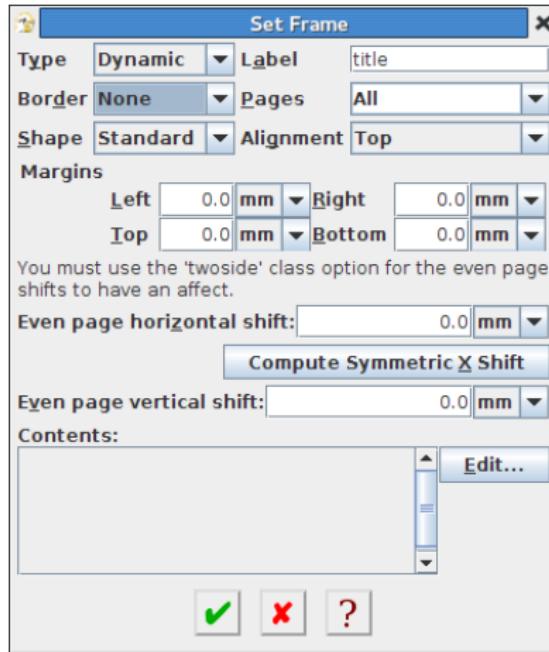


Figure 10.20 FlowframTk — Assigning Dynamic Frame Data

10.5 The *flowfram* Package and the *flowframtk* Application

At the bottom of this dialog window is an area labelled Contents with a button labelled Edit next to it. This allows you to set the contents of a dynamic or static frame, just as you can do using commands such as `\setdynamiccontents`, described earlier. Click on this Edit button to add the frame contents, which is the \LaTeX code from [Example 54](#):



\raggedright\bfseries\scshape\Large
Oh No! The Chickens Have Escaped

↑ Input

↓ Input

As shown in [Figure 10.21](#). Click on the green tick button to save these changes and return to the previous dialog window, and click on the green tick button there to save and close that window.

Links to bitmap images can be included in *flowframtk* images and they can also be used as frame backgrounds. Note that only the bitmap location is saved in an .ajr or .jdr file. You can choose whether this location is an absolute path or a path relative to the .ajr/.jdr file. To do this, use the Settings→Configure Image Settings menu item to open the image settings dialog window (shown earlier in [Figure 10.9](#)). Select the Bitmaps tab and either check or uncheck the Use relative paths for bitmaps button. You can also use this panel to select which \LaTeX command to use for inserting images. In [Figure 10.22](#) I have set this to `\includegraphics`.

10.5 The flowfram Package and the flowframtk Application

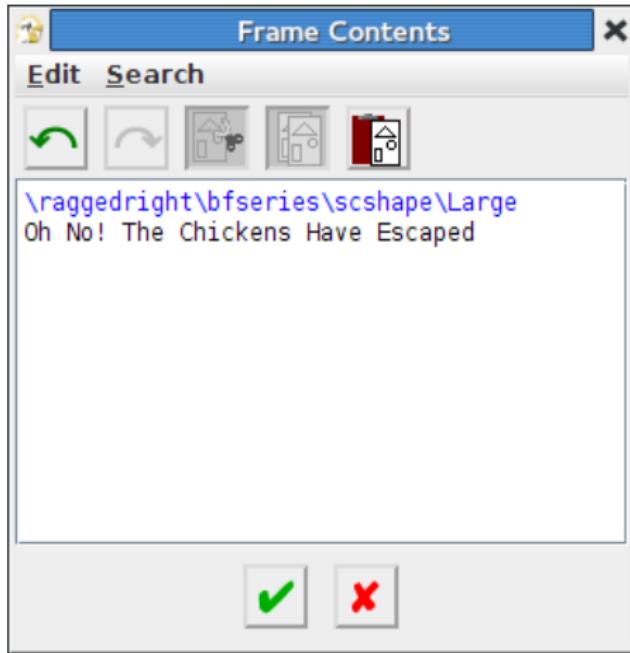


Figure 10.21 FlowframTk — Setting the Frame Contents

10.5 The *flowfram* Package and the *flowframtk* Application

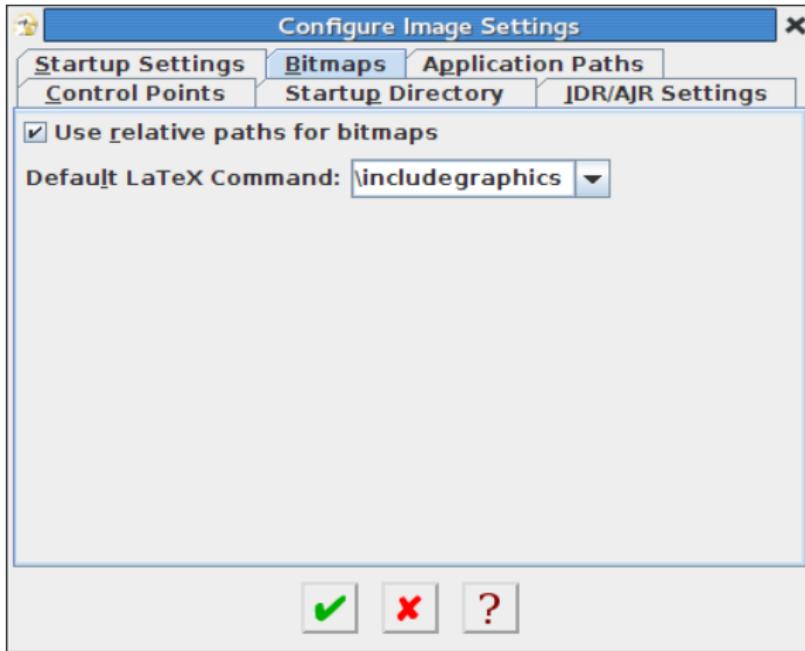


Figure 10.22 FlowframTk — Bitmap Options

10.5 The *flowfram* Package and the *flowframtk* Application

To insert a bitmap, make sure you are in select mode (Tools→Select) and use the menu item Bitmap→Insert Bitmap to open the bitmap selector, shown in [Figure 10.23](#).

Select the required bitmap (I've chosen the sample image `chicken.png`) and click on the Open button. This will insert the bitmap into the image, but this particular image is far too big so it needs to be scaled. Note that, just like the transformation options in `\includegraphics`, this scaling doesn't modify the actual bitmap file. The newly inserted bitmap should already be selected. To scale it use the Transform→Scale menu item to open the scaling dialog window and set the scale factor, as shown in [Figure 10.24](#). I've set the scale factor to 0.1.

The bitmap is now the correct size but is in the wrong position, as new bitmaps are always inserted with the top left corner at the origin. With the select tool set, you can use the mouse to drag the bitmap to the desired location, as shown in [Figure 10.25](#).

This bitmap can be set as the background to a static frame in a similar manner to the earlier static frame (labelled "sidepane"). So make sure the bitmap is selected, use the TeX/LaTeX→Flow Frames→Set Frame menu item to open the dialog box, and set the Type to Static, the label to, say, "titleimage" and make sure the Border is set to As Shown. I repeated this process for the sample image `dummy-logo.png` (with the scale set to 0.2) and labelled this frame "logo". The image so far, with both bitmaps, is as shown in

10.5 The *flowfram* Package and the *flowframtk* Application

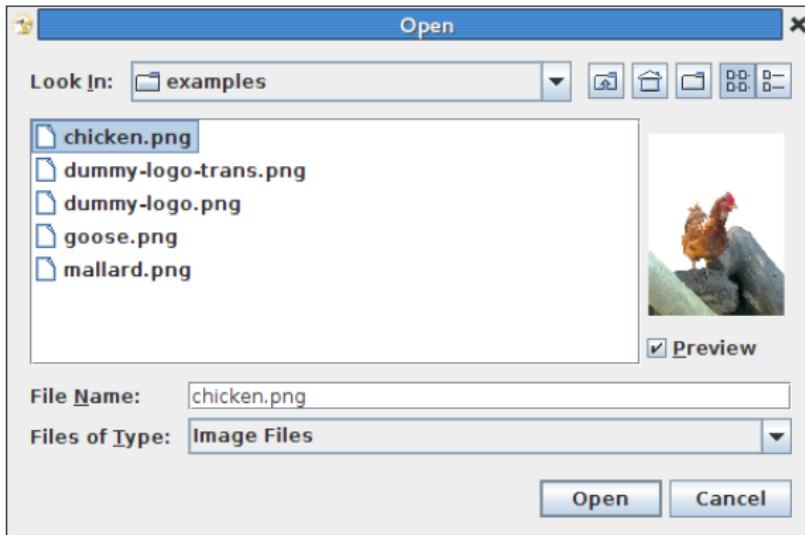


Figure 10.23 FlowframTk — Bitmap Selector

10.5 The *flowfram* Package and the *flowframtk* Application

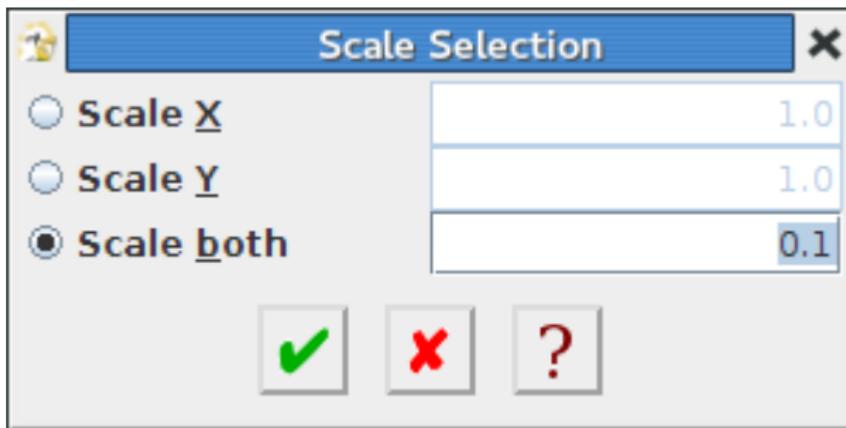


Figure 10.24 FlowframTk — Scaling

10.5 The *flowfram* Package and the *flowframtk* Application

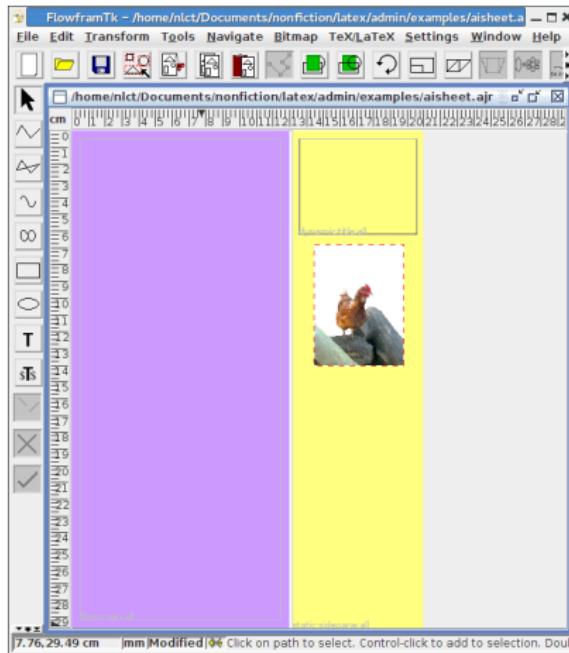


Figure 10.25 FlowframTk — Bitmap Moved to the Right

10.5 The *flowfram* Package and the *flowframtk* Application

Figure 10.26.

Now I just need another dynamic frame for the rest of the side panel information. Again this is done by creating a rectangle, selecting it and using the TeX/LaTeX→Flow Frames→Set Frame menu item to open the frame dialog window. Here I've set the Type to Dynamic, the label to "bookdata" and set Border to None. I've changed the Alignment option to Middle which will vertically balance the frame's contents. The contents can again be set by clicking on the Edit button, which will open the mini-LATEX editor. The contents are as follows:

```
\begin{tabular}{@{}l} \\[1mm] \begin{array}{l} \text{Genre:} & \& \text{Children's Illustrated} \\ & \& \text{Fiction} \\ \text{RRP:} & \& \text{\textsterling}5.99 \\ \text{Format:} & \& \text{Paperback} \\ \text{Pages:} & \& 30 \\ \text{Pub Date:} & \& \text{1st August 2014} \\ \text{ISBN:} & \& \text{978-x-xxxxxx-xx-x} \end{array} \\[1mm] \end{tabular} \\[3mm] \bigskip
```

↑ Input

10.5 The *flowfram* Package and the *flowframtk* Application

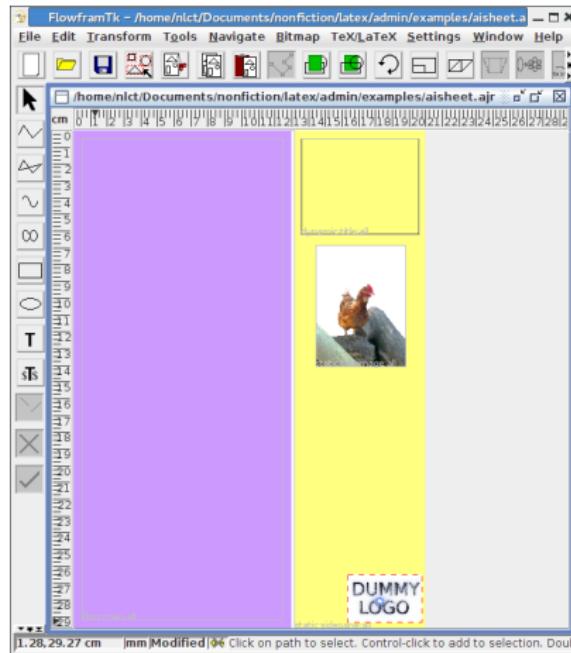


Figure 10.26 FlowframTk — Logo Added

10.5 The *flowfram* Package and the *flowframtk* Application

```
\begin{center}
\colorbox{white}{\includegraphics{barcode-qr}}
\end{center}
```

↓ Input

(If you prefer, you could convert `barcode-qr.pdf` to a bitmap and insert it in a similar manner to the other bitmaps.) Click the green tick to save and return to the Set Frame dialog, shown in [Figure 10.27](#).

The final image is as shown in [Figure 10.28](#). Make sure you save it to an `.ajr` or `.jdr` before proceeding to the export function. The `flowframtk` application can only load its own native files. It can't load the files it exports, so if you need to make any modifications you'll need the `.jdr/.ajr` file.

Now that the image is complete with all the required `flowfram` data, you can export to a L^AT_EX class or package file using the File→Export menu item. I'm going to export to a class file, so I've set the file type filter to Class (*.cls) as shown in [Figure 10.29](#).

My examples directory now contains the files: `aisheet.ajr`, `aisheet.cls` along with my original image files `chicken.png`, `barcode-qr.pdf` and `dummy-logo.png`. Now I just need to add a L^AT_EX document that uses this new `aisheet` class file:

↑ Input

10.5 The *flowfram* Package and the *flowframtk* Application

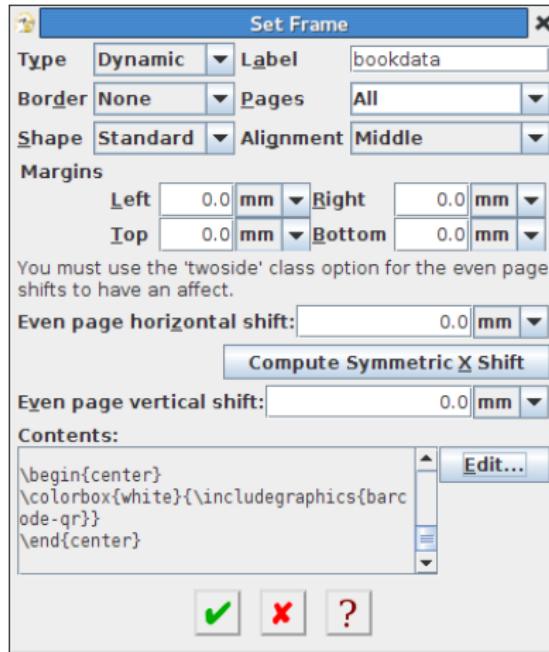


Figure 10.27 FlowframTk — Setting the bookdata Frame

10.5 The *flowfram* Package and the *flowframtk* Application

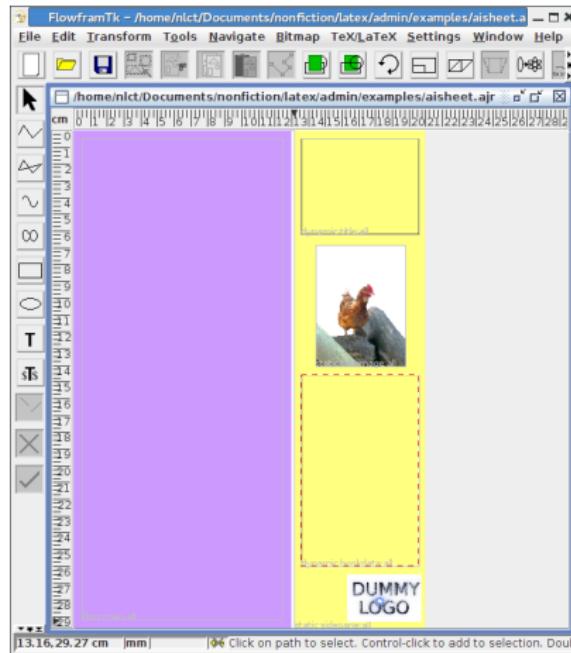


Figure 10.28 FlowframTk — Data Completed

10.5 The *flowfram* Package and the *flowframtk* Application

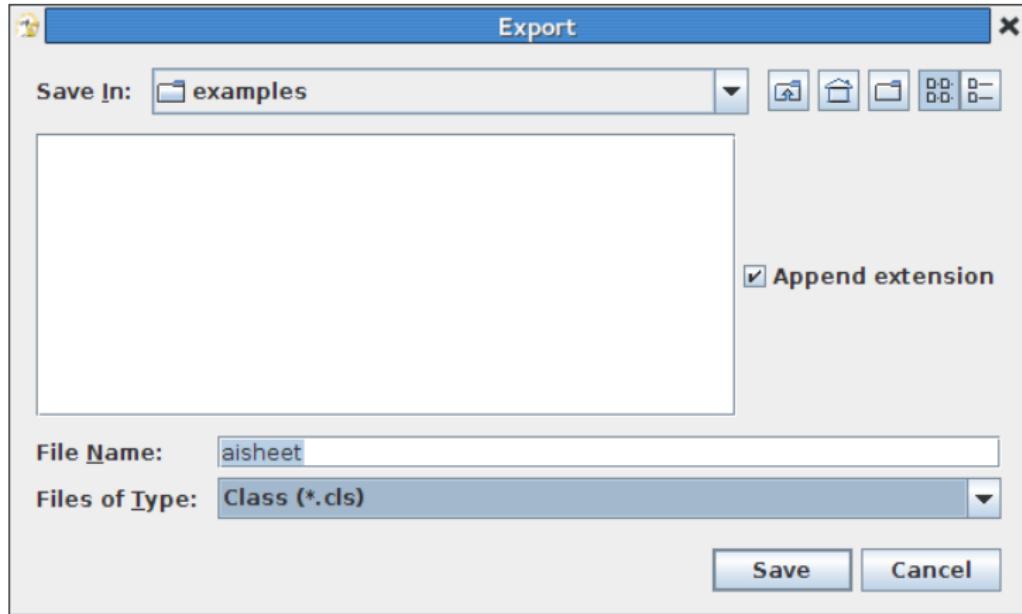


Figure 10.29 FlowframTk — Export to a Class File

10.5 The *flowfram* Package and the *flowframtk* Application

```
\documentclass{aisheet}
\begin{document}\raggedright

\section{About the Book}
```

A fun illustrated children's story about some escaped chickens. Fred and Mabel are looking after Granny's chickens for the day but, oh no, they've escaped. Will Fred and Mabel find them all before the chickens get into the road or get eaten by the hungry fox?

```
\section{About the Author}
```

Dickie Duck lives somewhere or other and won the best fowl book award in 2014. He likes writing silly stories about ducks and chickens.

```
\section{Keypoints}
```

```
\begin{dinglist}{118}
```

\item A fun way of teaching children to count.

\item Children will enjoy the repetition and rhyme.

10.5 The *flowfram* Package and the *flowframtk* Application

```
\item Features chickens doing stupid things.  
  
\item Completely fictitious book encourages children's  
imagination.  
\end{dinglist}  
  
\section{Marketing}  
  
\begin{dinglist}{118}  
 \item Written by award-winning author.  
  
 \item Illustrated by world famous artist.  
  
 \item Some other really interesting marketing information.  
\end{dinglist}  
  
\section{Contact}  
  
\begin{tabular}{@{}l}  
Dickie Duck\\  
1 The Street\\
```

10.5 The *flowfram* Package and the *flowframtk* Application

```
Another Village\\
Some City\\
Imagineshire\\
YZ1 2AB
\end{tabular}
\end{document}
```

↓ Input

The resulting document is shown in Figure 10.30. You can [download](#) or [view](#) this example document.



10.5 The *flowfram* Package and the *flowframtk* Application

About the Book

A fun illustrated children's story about some escaped chickens. Fred and Mabel are looking after Grandpa's chickens for the day but oh no, they've escaped! Will Fred and Mabel find them all before the chickens get into the road or get eaten by the hungry fox?

About the Author

Dickie Duck lives somewhere or other and won the best local book award in 2014. He likes writing silly stories about ducks and chickens.

Keypoints

- ♦ A fun way of teaching children to count.
- ♦ Children will enjoy the repetition and rhyme.
- ♦ Features chickens doing stupid things.
- ♦ Completely fictitious book encourages children's imagination.

Marketing

- ♦ Written by award-winning author.
- ♦ Illustrated by world famous artist.
- ♦ Some other really interesting marketing information.

Contact

Dickie Duck
1, The Street
An offbeat Village
Some City
Somewhere Else
YZ1 1AB

OH NO! THE CHICKENS HAVE ESCAPED



Genre: Children's Illustrated Fiction
RRP: £5.99
Format: Paperback
Published: 1st August 2014
ISBN: 978-1-xxxxxx-xx-x



DUMMY LOGO

Figure 10.30 Advance Information Sheet (via *flowframtk*)

11. ■ FORMS

There aren't very many entries on CTAN that deal with forms. At the time of writing there are four entries listed in the [form-fillin topic](#) and only one of them, [formular](#) [115], is in both MiK $\mathrm{T}\mkern-1mu\mathrm{E}\mathrm{X}$ and T $\mathrm{E}\mathrm{X}$ Live. If you are interested in writing proposals, there is also a [proposal topic](#) but again there aren't many entries in it. Some of the exam/assignment classes or packages (see §9 [Assignments and Examinations](#)) that have multiple choice or fill-in-the-blank options could also be used to create forms.

As with leaflets and flyers, forms don't really conform to standard typesetting styles. Small forms, such as the one for contact details in [Example 51](#), can be created using [tabular](#)-like environments with `\hrulefill` or `\dotfill` for ruled or dotted line areas. There are font packages available that provide tick and cross symbols [65], such as [pifont](#) [84] and [wasy sym](#) [42]. The [decoration topic](#) includes packages, such as [framed](#) [3] or [mdframed](#) [20], that can place frames around regions of text.

The code for the query form from [Example 51](#) is reproduced in the document below. This just uses the standard `article` class instead of the `leaflet` class used in that example:

```
\documentclass{article}

\usepackage{wasysym}

\begin{document}
\section{Query Form}
```

If you'd like to know more about the exciting collaboration between the Secret Lab of Experimental Stuff and the Department of Stripy Confectioners please fill in your details below and post this slip to:

```
\bigskip

\begin{tabular}{@{}l}
Miss Ingperson\\
Secret Lab of Experimental Stuff\\
University of Somewhere\\
Some City\\
AB3 4YZ
\end{tabular}
```

\bigskip

\Square__I would like to receive quarterly newsletters.

\Square__I agree to having my memory wiped.

\Square__Yes, I'd really like to feed the ducks.

\bigskip

\begin{tabular}{@{}lp{4cm}}}

Name: & \dotfill \\

Address: & \dotfill\\

& \dotfill \\

& \dotfill \\

& \dotfill \\

Postcode: & \dotfill\\

Country: & \dotfill\\

Telephone: & \dotfill\\

Mobile: & \dotfill\\

Email: & \dotfill

```
\end{tabular}
```

```
\end{document}
```

↓ Input

Both `\dotfill` and `\hrulefill` are *leaders* [46] and fill the available horizontal space. In the above example, I used the column identifier `p{4cm}` to create a column of width 4 cm, which gives `\dotfill` 4 cm of horizontal space to fill. If I'd just used the `l` left alignment column identifier then no dotted line would have appeared.

EXAMPLE

```
Some text\hrulefill Some more text.
```

↑ Input

```
Some text\dotfill Some more text.
```

```
\hrulefill Some text\dotfill Some more text.\hrulefill
```

↓ Input

produces:

```
Some text..... Some more text.  
Some text ..... Some more text.  
..... Some text..... Some more text.
```

↑ Output

↓ Output

You can place a leader inside a fixed-width box. For example:

```
The \makebox[3em]{\hrulefill} sat on the  
\makebox[3em]{\hrulefill}.
```

Input

which produces:

The _____ sat on the _____.

Output

A cut line can be produced with a combination of leaders and a pair of scissors symbol from a package such as pifont. For example:

```
\par\noindent  
\makebox[2em]{\dotfill}\ding{33}\dotfill\par
```

↑ Input

↓ Input

which produces:

.....

Output

Note that this doesn't extend the cut line into the margins. To achieve that you need to use some negative length and a horizontal box with width given by `\paperwidth`. For example:

```
\par\noindent
\hskip*{-\dimexpr 1in+\hoffset+\oddsidemargin}%
\rllap{%
  \makebox[\paperwidth][1]{%
    \makebox[4em]{\dotfill}\ding{33}\dotfill
  }}\par
```

↑ Input

↓ Input

For a two-sided document you will need to check if the current page is odd or even. For example:

```
\par\noindent
\ifodd\value{page}\relax
```

↑ Input

```
\hspace*{-.\\dimexpr 1in+\\hoffset+\\oddsidemargin}%
\\else
  \\hspace*{-.\\dimexpr 1in+\\hoffset+\\evensidemargin}%
\\fi
\\rlap{%
  \\makebox[\\paperwidth][l]{%
    \\makebox[4em]{\\dotfill}\\ding{33}\\dotfill
  }}\\par
```

↓ Input

 Be careful using this method of testing for an odd or even page, as it may not always work due to TeX's asynchronous output routine. If you are using one of the KOMA-Script classes, you can use KOMA-Script's `\ifthispageodd` command to determine if the current page is odd or even in a more robust manner.

[FAQ: Finding if
you're on an odd
or an even page]

You may find it easier to define a command that produces this with an optional argument to determine the distance between the start of the line and the scissor symbol. For example:

```
\\newcommand{\\cutline}[1][4em]{%
  \\par\\noindent
```

↑ Input

```
\ifodd\value{page}\relax
  \hspace*{-\dimexpr 1in+\hoffset+\oddsidemargin}%
\else
  \hspace*{-\dimexpr 1in+\hoffset+\evensidemargin}%
\fi
\rlap{%
  \makebox[\paperwidth][l]{%
    \makebox[#1]{\dotfill}\ding{33}\dotfill
  }}\par
}
```

↓ Input

Now you can just use this command, for example:

```
\cutline[6em]
```

Input



Remember that if you want a hard copy this requires borderless printing otherwise a slim margin may still appear (as occurs in the paperback version of this book).

If you are using the flowfram package, you also need to take into account any additional offset caused by a frame that doesn't have its left edge flush against the left edge of the typeblock. In this case the definition of \cutline needs to be adjusted as follows:

↑ Input

```
\newlength\frameoffset

\newcommand{\cutline}[1][4em]{%
  \par\noindent
  \ifodd\value{page}\relax
    \computeleftedgeodd{\frameoffset}%
    \getflowbounds{\value{thisframe}}%
  \else
    \computeleftedgeeven{\frameoffset}%
    \getflowevenbounds{\value{thisframe}}%
  \fi
  \addtolength{\frameoffset}{-\ffareax}%
  \hspace*{\frameoffset}%
  \rlap{%
    \makebox[\paperwidth][1]{%
      \makebox[#1]{\dotfill}\ding{33}\dotfill
    }}\par
}
```

↓ Input

11.1 Writing a Class File for a Form

EXERCISE 29. QUERY FORM

Reproduce the earlier query form from [Example 51](#) as a single-paged document (for example, using article or scrartcl) with no page numbering and a cut line between the submission address and the actual form.

You can [download](#) or [view](#) a solution to this exercise.

11.1 Writing a Class File for a Form

The [above](#) may be suitable for a short form to be filled in by hand, but it may be that you want to produce a more complex form to be filled in by L^AT_EX users. In this case, it may be more appropriate to write a class file that provides commands to fill in the form data. This section describes how to do this and is developed from an article I wrote on the L^AT_EX Community Forum [\[92\]](#). The [next section](#) will look at interactive form elements.

[§7.3](#) briefly introduced package writing. There are similar commands for classes, and some of the package commands, such as `\RequirePackage`, may also be used in class files. As with packages, the class first identifies the T_EX format using

11.1 Writing a Class File for a Form

`\NeedsTeXFormat{<format>}[<version>]`

Definition

and then identifies the class using

`\ProvidesClass{<name>}[<version>]`

Definition

This has the same syntax as `\ProvidesPackage` described in §7.3. The class code should be saved in a file called `<name>.cls` and placed somewhere on TeX's path.

Many classes load a parent class, which saves defining many common elements, such as the sectioning commands or list environments. The parent class is loaded using:

`\LoadClass[<options>]{<name>}[<version>]`

Definition

where `<name>` is the name of the parent class. The optional arguments are analogous to the optional arguments of `\RequirePackage`. Before you load a class, you can specify which options to pass to it using:

`\PassOptionsToClass{<option-list>}{<class-name>}`

Definition

where `<option-list>` is a [comma-separated list](#) of options to pass to the class specified by `<class-name>`. An option is defined using:

11.1 Writing a Class File for a Form

\DeclareOption{<option>}{<code>}

Definition

where <option> is the option name and <code> is the code to perform for that option. The starred version of this command only has one argument:

\DeclareOption*{<code>}

Definition

This indicates the code to perform for an unknown option. The option name can be referenced within <code> using

\CurrentOption

Definition

Once all the options have been declared, they then need to be processed using:

\ProcessOptions

Definition

Here's the code for a trivial class called simple-form:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{simple-form}[2014/10/11]

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
```

↑ Input

11.1 Writing a Class File for a Form

```
\ProcessOptions  
  
\LoadClass{article}  
  
\% class code  
  
\endinput
```

↓ Input

This code needs to be saved in a file called `simple-form.cls`. At the moment this class doesn't provide anything in addition to the article class, but the new code will be added in the area between the `\LoadClass` line and the `\endinput` line.

In addition to `\Square`, which produces an empty square \square , the `wasysym` package also defines

`\XBox`

Definition

which produces a box with a cross in it \boxtimes , and

`\CheckedBox`

Definition

which produces a box with a tick in it \boxdot . These symbols will be useful for this new class, so the class code needs to load the `wasysym` package using

11.1 Writing a Class File for a Form

```
\RequirePackage{wasysym}
```

Input

Information for the form can be gathered using the same type of mechanism as `\author`, `\title` and `\date`. These work by having an internal command that stores the information and a user command that sets the internal command. For example, if the form requires a person's name, the internal command could be called, say, `\@name` which is initially empty

```
\newcommand*{\@name}{}{}
```

Input

and the user command could be called, say, `\name` which redefines the internal command:

```
\newcommand*{\name}[1]{%
  \renewcommand*{\@name}{#1}%
}
```

↑ Input

↓ Input

Then a command analogous to `\maketitle` is required to typeset the form. For example, this command could be called `\makeform` and it would use the internal commands to fill in the required areas. A trivial example would be:

11.1 Writing a Class File for a Form

```
\newcommand{\makeform}{%
  Name: \@name\ Date: \@date
}
```

↑ Input

(\@date is the internal command used by \date and is initially defined as \today.)

This definition of \makeform has a problem when \name isn't used. If \@name is empty it won't take up any space. A better solution is to put \@name inside a horizontal box with a fixed width:

```
\newcommand{\makeform}{%
  Name: \makebox[6em][l]{\@name}\ Date: \@date
}
```

↑ Input

This will leave a blank space if the name hasn't been set. If you prefer a lined space you could make the initial definition of \@name use \hrulefill

11.1 Writing a Class File for a Form

```
\newcommand*{\@name}{\hrulefill}
```

Input

Now a lined space will appear if \name hasn't been used, but the line won't be present if \name has been used. If you still want a line to appear even if \name has been used, then you could replace

```
\makebox[6em][l]{\@name}
```

Input

with

```
\makebox[6em][l]{\rlap{\@name}\hrulefill}
```

Input

If you have more than one blank area to fill in, then it's best to define a command to do this. For example:

```
\newcommand*{\form@fillin}[2]{%
  \makebox[#1][l]{\rlap{#2}\hrulefill}%
}
```

↑ Input

↓ Input

This has the syntax

11.1 Writing a Class File for a Form

```
\form@fillin{<width>}{{<text>}}
```

Definition

so the trivial definition of `\makeform` can now look something like:

```
\newcommand{\makeform}{%
  Name: \form@fillin{6em}{\@name}\_
  Date: \form@fillin{4em}{\@date}%
}
```

↑ Input

↓ Input

Variations of `\form@fillin` could include

```
\newcommand*{\form@fillin}[2]{%
  \makebox[#1][l]{\rlap{#2}\dotfill}}
```

↑ Input

↓ Input

which uses a dotted line instead or

11.1 Writing a Class File for a Form

↑ Input

```
\newcommand*{\form@fillin}[2]{%
  \makebox[#1][c]{%
    \hrulefill\makebox[0pt][c]{#2}\hrulefill}%
}
```

↓ Input

which centres the text within the ruled area or

↑ Input

```
\newcommand*{\form@fillin}[2]{%
  \makebox[#1][r]{\hrulefill\llap{#2}}%
}
```

↓ Input

which right-aligns the text within the ruled area.

Check boxes require a different interface, but there are various methods you can use. For example, for a gender check box you might want a command called, say, `\male` that ticks the “male” box and a command called, say, `\female` that ticks the “female” box. Alternatively you might prefer a command called, say, `\gender` that takes an argument which can

11.1 Writing a Class File for a Form

either be male or female. In both cases, internal commands are defined for each option that default to the unchecked case:

```
\newcommand*{\gender@male}{\Square}  
\newcommand*{\gender@female}{\Square}
```

↑ Input

↓ Input

The user commands redefine these internal commands. In the first case:

```
\newcommand*{\male}{%  
  \renewcommand*{\gender@male}{\XBox} %  
}  
\newcommand*{\female}{%  
  \renewcommand*{\gender@female}{\XBox} %  
}
```

↑ Input

↓ Input

In the second case:

11.1 Writing a Class File for a Form

↑ Input

```
\newcommand*{\gender}[1]{%
  \ifcsdef{gender@#1}%
  {\csdef{gender@#1}{\XBox}}
  {% unknown option produces an error
    \ClassError{simple-form}{Unknown gender `#1'}
      {Options: `male', `female'}
  }%
}
```

↓ Input

This uses the etoolbox commands `\ifcsdef` and `\csdef` described in §2.1.1, and also uses

`\ClassError{<class-name>}{<error-message>}{<help-message>}`

Definition

to display an error message. The first argument is the class name (`simple-form` in this case) and the second argument is the error message. The third argument provides a help message if the user types “h” in TeX’s interactive mode.

What if I later decide to use `\CheckBox` instead of `\XBox`? Alternatively, I might decide to use a radio button style. To help with code maintenance

11.1 Writing a Class File for a Form

it's better to define commands for the checked and unchecked status and use those commands for the form data. For example:

```
\newcommand*{\form@unchecked}{\Square}
\newcommand*{\form@checked}{\XBox}

\newcommand*{\gender@male}{\form@unchecked}
\newcommand*{\gender@female}{\form@unchecked}

\newcommand*{\male}{%
    \renewcommand*{\gender@male}{\form@checked}%
}
\newcommand*{\female}{%
    \renewcommand*{\gender@female}{\form@checked}%
}
```

↑ Input

↓ Input

Or

```
\newcommand*{\gender}[1]{%
```

↑ Input

11.1 Writing a Class File for a Form

```
\ifcsdef{gender@#1}%
{\csdef{gender@#1}{\form@checked}}
{%
  \ClassError{simple-form}{Unknown gender `#1'}
  {Options: `male', `female'}%
}
}
```

↓ Input

Now there are only one or two lines to change if I want to use different symbols. For example, to use `\CheckBox` instead of `\XBox` just requires one edit:

```
\newcommand*{\form@checked}{\CheckBox}
```

Input

If you can't find a symbol that suits you, it's possible to combine symbols using a command such as `\rlap`. For example, to make round radio style buttons, you could use the `ifsym` package [45] with the `geometry` option and combine `\BigCircle` with `\FilledSmallCircle`.

```
\newcommand*{\form@unchecked}{\BigCircle}
\newcommand*{\form@checked}{\rlap{\FilledSmallCircle}\BigCircle}
```

↑ Input

↓ Input

11.1 Writing a Class File for a Form

These produce the symbols \bigcirc and \bullet .

⚠ Take care if you want to load both `ifsym` and `wasysym` as they have conflicting command names when `ifsym` is loaded with the `geometry` option. For example, both define `\Square`. If you want both packages, load `ifsym` without the `geometry` option and use `\text{ifsymbol}` to access the symbols. For example:

```
\newcommand*{\form@checked}{%
  \rlap{\text{ifsymbol}[ifgeo]{117}}\text{ifsymbol}[ifgeo]{37}}
\newcommand*{\form@unchecked}{\text{ifsymbol}[ifgeo]{37}}
```

↑ Input

Alternatively, if you want fancier buttons you can use picture drawing code. The following example creates on and off buttons using `tikz` with the `shadings` and `shadows` libraries:

```
\RequirePackage{x11names}{xcolor}
\RequirePackage{tikz}
\usetikzlibrary{shadings}
\usetikzlibrary{shadows}
```

↑ Input

11.1 Writing a Class File for a Form

```
\newcommand*{\form@unchecked}{%
  \resizebox{!}{2ex}%
{%
  \begin{tikzpicture}
    \path[fill=LightYellow4,circular glow] (0,0) circle(.5cm);
    \path[fill=LightYellow1,circular glow={fill=LightYellow3}]
      (0,0) circle(.35cm);
  \end{tikzpicture}%
}%
}

\newcommand*{\form@checked}{%
  \resizebox{!}{2ex}%
{%
  \begin{tikzpicture}
    \path[shade,inner color=LightYellow2,
          outer color=LightYellow4,
          circular glow] (0,0) circle(.5cm);
  \end{tikzpicture}%
}%
}
```

↓ Input

11.1 Writing a Class File for a Form

This produces and .

Similarly, it's a good idea to provide a command to layout the check box or fill-in area and its accompanying text. For example:

```
\newcommand*{\form@layout@checkbox}[2]{#1 #2}
```

Input

This has the syntax:

```
\form@layout@checkbox{<symbol>}{<text>}
```

Definition

For example:

```
\form@layout@checkbox{\gender@male}{Male}  
\form@layout@checkbox{\gender@female}{Female}
```

↑ Input

↓ Input

This means that if, say, you want to change all your check boxes so that the text is to the left of the check box symbol, then all you need to do is change the definition of \form@layout@checkbox. Similarly for the fill-in text fields:

```
\newcommand*{\form@layout@fillin}[3]{\form@fillin{#1}{#2}}
```

Input

This has the syntax

11.1 Writing a Class File for a Form

```
\form@layout@fillin{\width}{\value}{\text}
```

Definition

For example:

```
\form@layout@fillin{6em}{@\name}{Name}
```

Input

EXAMPLE 56. A SIMPLE FORM CLASS

Here's a simple form class with two fill-in areas (for the name and date) and two check boxes (for the gender). The article class is loaded with the options a4paper and 12pt as this example is simulating a form with specific paper size and font requirements.

The contents of the file `simple-form.cls` are as follows:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{simple-form}[2014/10/11]

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions

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

↑ Input

11.1 Writing a Class File for a Form

```
\RequirePackage{etoolbox}
\RequirePackage{wasysym}

\newcommand*\form@fillin}[2]{%
  \makebox[#1][l]{\rlap{#2}\hrulefill}%
}

\newcommand*\form@checked}{\XBox}
\newcommand*\form@unchecked}{\Square}

\newcommand*\form@layout@checkbox}[2]{#1 #2}
\newcommand*\form@layout@fillin}[3]{#3: \form@fillin{#1}{#2} }

\newcommand*\@name{}
\newcommand*\name}[1]{\renewcommand*\@name{#1} }

\newcommand*\gender@male}{\form@unchecked}
\newcommand*\gender@female}{\form@unchecked}

\newcommand*\gender}[1]{%
  \ifcsdef{gender@#1}{%
```

11.1 Writing a Class File for a Form

```
{\csdef{gender@#1}{\form@checked}}
{%
  \ClassError{simple-form}{Unknown gender `#1'}%
  {Options: `male', `female'}%
}
}

\newcommand{\makeform}{%
  \form@layout@fillin{8em}{\@name}{Name}\qquad
  \form@layout@fillin{12em}{\@date}{Date}
  \par
  \bigskip
  \par
  \form@layout@checkbox{\gender@male}{Male}\qquad
  \form@layout@checkbox{\gender@female}{Female}
}

\endinput
```

↓ Input

An example document:

↑ Input

11.1 Writing a Class File for a Form

```
\documentclass{simple-form}

\name{Mabel Canary}
\gender{female}

\begin{document}

\makeform

\end{document}
```

↓ Input

The result is shown in Figure 11.1. You can [download](#) or [view](#) this example.

Name: Mabel Canary _____ Date: October 12, 2014 _____

Output

Male Female

Figure 11.1 A Simple Form with Two Fill-In Areas and Two Check Boxes

11.1 Writing a Class File for a Form

The above example doesn't test if \gender has already been used, so it's possible for a user to do:

```
\gender{male}\gender{female}
```

Input

which would cause both boxes to be checked. If you want to prevent this from happening you could either produce an error message if the command is used more than once or make each subsequent use of the command reset the boxes before setting the new choice.

Here's a possible way of implementing the first case. It uses the \let assignment described in §2.1.1.

```
\newcommand*{\@gendererror}[1]{%
  \ClassError{simple-form}
  {\string\gender\space may only be used once}
  {}%
}

\newcommand*{\gender}[1]{%
  \let\gender\@gendererror
  \ifcsdef{gender@#1}%
  {\csdef{gender@#1}{\form@checked}}
```

↑ Input

11.1 Writing a Class File for a Form

```
{%
  \ClassError{simple-form}{Unknown gender `#1'}%
  {Options: `male', `female'}%
}%
}
```

↓ Input

This works as follows: the first time `\gender` is used, it redefines itself to have the same definition as `\@gendererror`, so the next time `\gender` is used, it's now equivalent to `\@gendererror`, which ignores its argument and produces an error message. (`\string` is a TeX primitive that converts the following control sequence into a list of characters, which provides an easy way of printing the control sequence in the transcript file or console.)

Here's a possible way of implementing the second case that defines a reset command:

```
\newcommand*{\@resetgender}{%
  \renewcommand*{\gender@male}{\form@unchecked}%
  \renewcommand*{\gender@female}{\form@unchecked}%
}
```

↑ Input

11.1 Writing a Class File for a Form

```
\newcommand*{\gender}[1]{%
  \@resetgender
  \ifcsdef{gender@#1}{%
    \csdef{gender@#1}{\form@checked}%
  }{%
    \ClassError{simple-form}{Unknown gender `#1'}{%
      Options: `male', `female'}%
  }%
}
```

↓ Input

The `\male/\female` version is simpler:

```
\newcommand*{\male}{%
  \renewcommand*{\gender@female}{\form@unchecked}
  \renewcommand*{\gender@male}{\form@checked}%
}

\newcommand*{\female}{%
  \renewcommand*{\gender@male}{\form@unchecked}
  \renewcommand*{\gender@female}{\form@checked}%
}
```

↑ Input

11.1 Writing a Class File for a Form

However the other method is neater for a large set of check boxes. If you do have many choices, you may find it easier to use a list-based approach. For example, suppose I want to produce the following:

Which project would you like to enrol on?

- Mind-Controlling Cookies Telepathic Cakes
- Exploding Chocolates Ray Gun

A convenient user command might be called, say, \project where the argument may be one of: cookies, cakes, chocolates or raygun. The internal commands are called \project@*<label>* where *<label>* is the argument of \project. These commands can be reset using:

```
\csdef{project@<label>}{\form@unchecked}
```

Input

and set using

```
\csdef{project@<label>}{\form@checked}
```

Input

These can be wrapped up in two commands that each take the label as the argument:

11.1 Writing a Class File for a Form

↑ Input

```
\newcommand*{\reset@project}[1]{%
  \csdef{project@#1}{\form@unchecked}%
}
\newcommand*{\set@project}[1]{%
  \ifcsdef{project@#1}
    {\csdef{project@#1}{\form@checked}}
    {%
      \ClassError{simple-form}{Unknown project `#1'}{}%
    }%
}
```

↓ Input

It's also useful to provide a command to use the internal `\project@<label>` command:

↑ Input

```
\newcommand*{\use@project}[1]{%
  \ifcsdef{project@#1}{\csuse{project@#1}}{\form@unchecked}%
}
```

↓ Input

11.1 Writing a Class File for a Form

This will produce an unchecked box if the label hasn't been defined, which means that the internal commands don't need to be initialised if the user wants a blank form to fill in by hand.

Here's a **comma-separated list** where each element contains two groups. The first is the label that will be used in the argument of \project and the second is the text to appear next to the check box in the form:

```
\newcommand*{\@projectlist}{%
  {cookies}{Mind-Controlling Cookies},%
  {cakes}{Telepathic Cakes},%
  {chocolates}{Exploding Chocolates},%
  {raygun}{Ray Gun}%
}
```

↑ Input

↓ Input

Various list-iteration commands were discussed in §2.7.2, but in the examples from that section all of the lists had an element that could be used as a single argument to a command such as \do. However in this list each element needs to be treated as two arguments. For example, the command to reset the check boxes should iterate through this list but only grab the first group (the label) of each element.

11.1 Writing a Class File for a Form

Consider first:

```
\@for>this@element:=@\projectlist\do{%
    \reset@project>this@element
}
```

↑ Input

✗
↓ Input

This won't work because it's equivalent to doing

```
\reset@project{{cookies}}{Mind-Controlling Cookies}
```

✗

and so on. I could try using `\expandafter` described in §2.7.2:

```
\@for>this@element:=@\projectlist\do{%
    \expandafter\reset@project>this@element
}
```

↑ Input

✗
↓ Input

11.1 Writing a Class File for a Form

This is an improvement as this is now equivalent to doing

```
\reset@project{cookies}{Mind-Controlling Cookies}
```



and so on. Now `\reset@project` picks up the label correctly, but the text after the label is left dangling and needs to be discarded. There are various ways to deal with this. The simplest solution is just to make `\reset@project` take two arguments and ignore the second argument:

```
\newcommand*{\reset@project}[2]{%
  \csdef{project@#1}{\form@unchecked}%
}
```

↑ Input

↓ Input

A more generic approach is to leave `\reset@project` with just one argument as before and use the L^AT_EX kernel command

```
@firstoftwo{\first}{\second}
```

Definition

which does `\first` and discards `\second`. This requires `\expandafter` to expand `\this@element` before applying `@firstoftwo`:

11.1 Writing a Class File for a Form

↑ Input

```
\@for\this@element:=\@projectlist\do{%
  \reset@project{\expandafter\@firstoftwo\this@element}%
}
```

↓ Input

A similar method can be used to display the check boxes and their associated text within the form. There is an analogous L^AT_EX kernel command that grabs the second argument and discards the first:

```
\@secondoftwo{\langle first\rangle}{\langle second\rangle}
```

Definition

Here's a simple example that just displays the check boxes with their associated text without any tabulation:

↑ Input

```
\@for\this@element:=\@projectlist\do{%
  \use@project{\expandafter\@firstoftwo\this@element}% check box
  \space
  \expandafter\@secondoftwo\this@element
  \qquad
}
```

↓ Input

11.1 Writing a Class File for a Form

Or using the layout command `\form@layout@checkbox` defined earlier:

```
↑ Input
@for\this@element:=@\projectlist\do{%
  \form@layout@checkbox
    {\use@project{\expandafter\@firstoftwo\this@element}}% check box
    {\expandafter\@secondoftwo\this@element}% text
  \qquad
}
↓ Input
```

This can be converted into a `tabular` environment but we need a way to track which column we're in. One way to do this is to define a register (recall §2.1.3).

```
% initialise
\newcount\form@columncount
\form@columncount=1\relax
\def\form@precolumn{}%
% layout check boxes and text:
\begin{tabular}{ll}
```

11.1 Writing a Class File for a Form

```
\@for\this@element:=\@projectlist\do{%
  \global\let\this@element\this@element
  \form@precolumn
  \form@layout@checkbox
    {\use@project{\expandafter\@firstoftwo\this@element}}%
    {\expandafter\@secondoftwo\this@element}%
  \global\advance\form@columncount by 1\relax
  \ifnum\form@columncount>2\relax
    \global\form@columncount=1\relax
    \gdef\form@precolumn{\\"}%
  \else
    \gdef\form@precolumn{\&}%
  \fi
}%
\end{tabular}
```

↓ Input

(`\global` is required because of the local scoping effect of tabular cells.)
This uses a similar method to those discussed in §2.7.5.

If you are likely to have more than one group of check boxes, then it makes more sense to create generic commands. First, we need generic versions of the above `\reset@project`, `\set@project` and `\use@project` where the first argument is the element label (such as `cakes`) and the

11.1 Writing a Class File for a Form

second argument is the block label (such as project):

```
\newcommand*{\reset@element}[2]{%
  \csdef{#2@#1}{\form@unchecked}%
}

\newcommand*{\set@element}[2]{%
  \ifcsdef{#2@#1}%
  {\csdef{#2@#1}{\form@checked}}%
  {%
    \ClassError{simple-form}{Unknown #2 `#1'}{}%
  }%
}

\newcommand*{\use@element}[2]{%
  \ifcsdef{#2@#1}{\csuse{#2@#1}}{\form@unchecked}%
}
```

↑ Input

↓ Input

So now instead of

11.1 Writing a Class File for a Form

```
\reset@project{\label}
```

Input

I need to use

```
\reset@element{\label}{project}
```

Input

and so on. It's also convenient to provide a command that can iterate over the {\label}{text} list (such as \@projectlist) for the block:

```
\newcommand*\for@block}[3]{%
\ifcsdef{@#2list}%
{%
\expandafter\@for\expandafter
#1\expandafter:\expandafter=\cscname @#2list\endcscname\do{#3}%
}%
{%
\ClassError{simple-form}{Unknown block `#2'}{}%
}%
}
```

↑ Input

↓ Input

11.1 Writing a Class File for a Form

(The `\expandafter`s are required because the list control sequence provided by `\csname @#2list\endcsname` needs to be expanded to the actual control sequence `\@<block-label>list`, for example `\@projectlist`, before `\@for` tries to iterate over it.) This has the syntax:

```
\for@block{\langle cs\rangle}{\langle block-label\rangle}{\langle body\rangle}
```

Definition

where `\langle cs\rangle` is assigned to the `\{\langle label\rangle\}\{\langle text\rangle\}` element for the current iteration.

All elements within a block can be reset using `\reset@block`, which is defined as:

```
\newcommand*\reset@block[1]{%
  \for@block{\this@element}{#1}{%
    %
      \reset@element{\expandafter\@firstoftwo\this@element}{#1}%
    %
  }
}
```

↑ Input

↓ Input

This means that `\project` can now be defined as

11.1 Writing a Class File for a Form

↑ Input

```
\newcommand*{\project}[1]{%
  \reset@block{project}%
  \set@element{#1}{project}%
}
```

↓ Input

The generic two-column tabulated block of elements used by `\makeform` can be defined as follows:

↑ Input

```
\newcount\form@columncount

\newcommand*{\form@block}[1]{%
  \def\form@precolumn{}%
  \form@columncount=1\relax
  \begin{tabular}{ll}
  \for@block{\this@element}{#1}%
  {%
    \global\let\this@element\this@element
    \form@precolumn
```

11.1 Writing a Class File for a Form

```
\form@layout@checkbox
  {\use@element{\expandafter\@firstoftwo\this@element}{#1}}%
  {\expandafter\@secondoftwo\this@element}%
\global\advance\form@columncount by 1\relax
\ifnum\form@columncount>2\relax
  \global\form@columncount=1\relax
  \gdef\form@precolumn{\\"}%
\else
  \gdef\form@precolumn{&}%
\fi
}%
\end{tabular}%
}
```

↓ Input

This custom command has the syntax:

```
\form@block{\<block-label>}
```

Definition

So for the project example, this would just require

```
\form@block{project}
```

Input

This is hard-coded for two columns, but it would be more flexible to allow an arbitrary number of columns. For example if the command had the

11.1 Writing a Class File for a Form

syntax

```
\form@block{\<block-label>}{\<columns>}
```

Definition

then the project block could be generated using

```
\form@block{project}{2}
```

Input

In this case, the hard-coded conditional in `\form@block`

```
\ifnum\form@columncount>2\relax
```

Input

can now have the total column count replaced with #2:

```
\ifnum\form@columncount>#2\relax
```

Input

However the column specifier argument for the `tabular` environment is a little more complicated as it now requires #2 lots of 1 (or whatever alignment specifier you want).

Recall TeX's `\loop` command from §2.7.4 and the hook management commands from §2.1.2. These can be used to generate the argument for the `tabular` environment:

```
% initialise
```

↑ Input

11.1 Writing a Class File for a Form

```
\def\form@columnargs{}%
\form@columncount=0\relax
% iterate #2 times
\loop
\appto\form@columnargs{1}%
\advance\form@columncount by 1\relax
\ifnum\form@columncount<#2
\repeat
```

↓ Input

This will store the column specifiers in `\form@columnargs` which can now be used in the `tabular` environment argument:

```
\begin{tabular}{\form@columnargs}
```

Input

Therefore the new two-argument version of `\form@block` can be defined as:

```
\newcount\form@columncount

\newcommand*\form@block}[2]{%
\def\form@columnargs{}%
```

↑ Input

11.1 Writing a Class File for a Form

```
\form@columncount=0\relax
\loop
  \appto\form@columnargs{1}%
  \advance\form@columncount by 1\relax
\ifnum\form@columncount<#2
\repeat
\def\form@precolumn{}%
\form@columncount=1\relax
\begin{tabular}{\form@columnargs}
\for@block\this@element{#1}%
{%
  \global\let\this@element\this@element
  \form@precolumn
  \form@layout@checkbox
    {\use@element{\expandafter\@firstoftwo\this@element}{#1}}%
    {\expandafter\@secondoftwo\this@element}%
\global\advance\form@columncount by 1\relax
\ifnum\form@columncount>#2\relax
  \global\form@columncount=1\relax
  \gdef\form@precolumn{\}%
\else
  \gdef\form@precolumn{&}%
```

11.1 Writing a Class File for a Form

```
\fi  
}%  
\end{tabular}  
}  
|
```

↓ Input

The form check box elements are now much simpler to define:

```
\newcommand*{\@genderlist}{\{male\}{Male},\{female\}{Female}}  
  
\newcommand*{\gender}[1]{%  
  \reset@block{gender} %  
  \set@element{#1}{gender} %  
}  
  
\newcommand*{\@projectlist}{%  
  \{cookies\}{Mind-Controlling Cookies},%  
  \{cakes\}{Telepathic Cakes},%  
  \{chocolates\}{Exploding Chocolates},%  
  \{raygun\}{Ray Gun} %  
}
```

↑ Input

11.1 Writing a Class File for a Form

```
\newcommand*{\project}[1]{%
  \reset@block{project}%
  \set@element{#1}{project}%
}
```

↓ Input

If multiple selections are permitted, then the `\reset@block` command needs to be moved outside the user command definition to initialise all the elements. For example, if multiple projects may be selected:

```
\reset@block{project}

\newcommand*{\project}[1]{%
  \set@element{#1}{project}%
}
```

↑ Input

↓ Input

EXERCISE 30. SIMPLE FORM CLASS WITH CHECK BOXES

Adapt the class file `simple-form.cls` from [Example 56](#) so that the form shown in [Figure 11.2](#) can be created with the following document:

11.1 Writing a Class File for a Form

↑ Input

```
\documentclass{simple-form}

\name{Mabel Canary}
\date{2014-10-13}

\gender{female}
\project{cakes}
\icecream{vanilla}
\icecream{fudge}
\icecream{other}

\begin{document}

\makeform

\end{document}
```

↓ Input

FOR THE MORE ADVENTUROUS

Add a fill-in area for the “Other” ice-cream option so that instead of the user writing:

11.1 Writing a Class File for a Form

```
\icecream{other}
```

Input

they can use a new command:

```
\othericecream{Neapolitan}
```

Input

which both checks the “Other” box and fills in the area, as shown in [Figure 11.3](#). You can [download](#) or [view](#) a solution to this exercise.

If you have a large text area that needs to be filled in, you may prefer to use an environment to collect the information. For example, instead of creating a command to specify, say, a project description:

```
\projectdescription{(Several paragraphs of text)}
```

Input

which can be defined using

```
\newcommand{\@projectdescription}{}  
\newcommand{\projectdescription}[1]{%  
  \renewcommand{\@projectdescription}{#1}%  
}
```

↑ Input

↓ Input

11.1 Writing a Class File for a Form

Output

Name: Mabel Canary

Date: 2014-10-13

Male Female

Which project would you like to enrol on? (Tick one box.)

- Mind-Controlling Cookies Telepathic Cakes
- Exploding Chocolates Ray Gun

Which ice-cream flavours do you like? (Tick all that apply.)

- Vanilla Mint Toffee
- Fudge Guaraná Strawberry
- Raspberry Ripple Chilli Other

Figure 11.2 A Simple Form with Multiple Check Box Areas

11.1 Writing a Class File for a Form

Output

Name: Mabel Canary _____ Date: 2014-10-13 _____

Male Female

Which project would you like to enrol on? (Tick one box.)

- Mind-Controlling Cookies Telepathic Cakes
- Exploding Chocolates Ray Gun

Which ice-cream flavours do you like? (Tick all that apply.)

- Vanilla Mint Toffee
- Fudge Guarana Strawberry
- Raspberry Ripple Chilli Other: Neapolitan _____

Figure 11.3 A Simple Form with Multiple Check Box Areas and a Fill-In Other Area

11.1 Writing a Class File for a Form

you may prefer to have the user interface:

```
\begin{ProjectDescription}  
<Several paragraphs of text>  
\end{ProjectDescription}
```

↑ Input

↓ Input

This is more complicated to define, as you can't simply gather the contents of an environment when you use `\newenvironment`. There are a number of ways to achieve this.

⚠ If the environment contents are being gathered so that they can then be stored in a command definition, then the limitations applied to command definitions also apply to the environment contents. This includes the usual problems with verbatim code in a command argument.

The `collect` package [77] provides the `collectinmacro` environment:

```
\begin{collectinmacro}{(macro)}{(before)}{(after)}  
(body)  
\end{collectinmacro}
```

[FAQ: Why
doesn't verbatim
work within...?]

Definition

This defines the command `(macro)` to be `(before)(body)(after)`. For example,

11.1 Writing a Class File for a Form

↑ Input

```
\begin{collectinmacro}{\mycommand}{Before. }{ After.}  
Some text here.  
\end{collectinmacro}
```

↓ Input

This is equivalent to:

```
\newcommand{\mycommand}{Before. Some text here. After.}
```

Input

If you want to define an environment that uses this method, you can't use the environment form `\begin{collectinmacro}` and `\end{collectinmacro}`, but must instead use the commands `\collectinmacro` and `\endcollectinmacro`.

EXAMPLE:

↑ Input

```
\newcommand{@projectdescription}{}  
\newenvironment{ProjectDescription} %  
{\collectinmacro{@projectdescription}{}{}%}  
\endcollectinmacro
```

↓ Input

11.1 Writing a Class File for a Form

Another possibility is to use the amsmath package's

`\collect@body\{cs}`

Definition

command. This gathers the contents of the current environment `\{body\}` and applies `\{cs\}\{body\}`.

EXAMPLE:

```
\newcommand{\@projectdescription}{}  
\newcommand{\projectdescription}[1]{%  
  \gdef\@projectdescription{\#1}%  
}  
\newenvironment{ProjectDescription}{%  
  \begin{list}{  
    \collect@body\projectdescription  
  }  
}
```

↑ Input

↓ Input

This now means that the user can do either

```
\begin{ProjectDescription}
```

↑ Input

11.1 Writing a Class File for a Form

This will be an interesting project.

```
\end{ProjectDescription}
```

↓ Input

or

```
\projectdescription{This will be an interesting project.}
```

Input

Note that I had to use `\gdef` instead of `\renewcommand` otherwise the change will be scoped by the encasing environment.

 The `\collect@body` command uses short internal commands to gather the environment contents, which means that the environment can't contain paragraph breaks. If you want to allow paragraph breaks, you can use an analogous command provided by the `environ` package [75]:

```
\Collect@Body{cs}
```

Definition

Note that the unstarred version of `\newcommand` allows a paragraph break to be present within #1 so `\projectdescription` can be used by `\Collect@Body` in the following:

```
\newcommand{\@projectdescription}{}  
\newcommand{\projectdescription}[1]{%
```

↑ Input

11.1 Writing a Class File for a Form

```
\gdef\@projectdescription{#1}%
}
\newenvironment{ProjectDescription}%
{\Collect@Body\projectdescription}%
{}
```

↓ Input

EXAMPLE 57. A SIMPLE FORM CLASS (GATHERING ENVIRONMENT CONTENTS)

This example uses the `\Collect@Body` command from the `environ` package to allow the user to enter multi-paragraph data in a form. First the class file, `sample-form.cls`:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{sample-form}[2014/11/03]

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions

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

↑ Input

11.1 Writing a Class File for a Form

```
\RequirePackage{environ}

\newcommand*{\form@fillin}[2]{%
  \makebox[#1][l]{\rlap{#2}\hrulefill}%
}

\newcommand*{\form@layout@fillin}[3]{#3: \form@fillin{#1}{#2} }

\newcommand*\@name{}

\newcommand*{\name}[1]{%
  \renewcommand*{\@name}{#1}%
}

\newcommand*\@projectdescription{}

\newcommand{\projectdescription}[1]{%
  \long\gdef\@projectdescription{#1}%
}

\newenvironment{ProjectDescription}{%
  \Collect@Body\projectdescription{}%
```

11.1 Writing a Class File for a Form

```
\newcommand{\makeform}{%
  \section{Applicant Details}
  \form@layout@fillin{8em}{\@name}{Name}
  \section{Project Description}
  \@projectdescription
}

\endinput
```

↓ Input

Here's an example document:

```
\documentclass{sample-form}

\name{Mabel Canary}

\begin{ProjectDescription}
This project will be very interesting.

This is another paragraph.
\end{ProjectDescription}
```

↑ Input

```
\begin{document}  
\makeform  
\end{document}
```

↓ Input

The result is shown in [Figure 11.4](#). You can [download](#) or [view](#) this example document.

11.2 ■ Electronic PDF Forms

The [previous section](#) just considered a PDF form that could be filled in using custom commands within the document. This section looks at creating a PDF form where a user can fill the form in using interactive buttons and text fields in a PDF viewer that supports forms, such as Adobe Reader.

The `hyperref` package [69] provides commands to generate an electronic PDF form. There's also the `eforms` package [91], which is part of the `AcroTeX` bundle, however this is only in `MiKTeX` and not in `TeX Live`. This section will look at using the `hyperref` package, since it's available in both `MiKTeX` and `TeX Live`.

Output

1 Applicant Details

Name: Mabel Canary

2 Project Description

This project will be very interesting.

This is another paragraph.

Figure 11.4 A Simple Form with a Text Area.

11.2 Electronic PDF Forms

The commands that generate the interactive elements of the form, must all be placed inside the `Form` environment.

```
\begin{Form}[\langle parameters\rangle]  
<form body>  
\end{Form}
```

Definition

The optional argument `\langle parameters\rangle` is a `key=value list` of options if a “submit” button is included in the form. Available options are:

`action` The value should be the URL to process the form data.

`encoding` The encoding of the URL. The norm is PDF-encoding. The only valid value for this option is `html`.

`method` Values can be `post` or `get`.

The interactive elements can be created within the `Form` environment using any of the following commands:

```
\TextField[\langle options\rangle]{\langle label\rangle}
```

Definition

to create a text field;

11.2 Electronic PDF Forms

`\CheckBox[⟨options⟩]{⟨label⟩}`

Definition

to create a check box;

`\ChoiceMenu[⟨options⟩]{⟨label⟩}{⟨choices⟩}`

Definition

to create a selection of choices, such as a list menu, a popup menu, a combo menu or a group of radio buttons, where `⟨choices⟩` is a [comma-separated list](#) of labels for each available choice or a [key=value list](#) of `⟨label⟩=⟨name⟩` options;

`\PushButton[⟨options⟩]{⟨label⟩}`

Definition

to create a push button;

`\Submit[⟨options⟩]{⟨label⟩}`

Definition

to create a submit button and

`\Reset[⟨options⟩]{⟨label⟩}`

Definition

to create a reset button.

In each case, the field has a textual label (given by `⟨label⟩`) and a [key=value list](#) of options. There are a large number of options available. For the full list, see the [hyperref manual \[69\]](#). A selection of common options follows:

11.2 Electronic PDF Forms

accesskey	Specifies the shortcut key to activate/focus an element. There is no default value.
align	Alignment within a text field. Allow values: 0 (left-aligned), 1 (centred), 2 (right-aligned). The default value is 0.
combo	A boolean key to indicate if the choice list is a combo menu. The default value is false.
default	The default value for a field.
hidden	A boolean key to indicate if the field is hidden. The default value is false.
menulength	The number of elements shown in a choice list. The default value is 4.
multiline	A boolean key to indicate if the text field is a multiline field.
name	The name of the field (defaults to the label if omitted). Note that the label is the text that appears by the side of the field (or on the button, in the case of a push button) whereas the name identifies the field when referenced in the script that processes the form.

11.2 Electronic PDF Forms

password	A boolean key to indicate if the text field is a password field. The default value is false .
popdown	A boolean key to indicate if the choice list is a popdown menu. The default value is false .
radio	A boolean key to indicate if the choice list is a group of radio buttons. The default value is false .
value	The initial value for the field.

EXAMPLE 58. A SIMPLE ELECTRONIC FORM

This example form doesn't have a submit button. Here, the user just fills in the form using the interactive elements and either prints it out or saves it. (The ability to save the PDF file depends on the PDF viewer, but if you aren't able to save it you may be able to print it to another PDF file, using a "print to file" option in your printer dialog.)

```
\documentclass{article}  
  
\usepackage{hyperref}
```

↑ Input

11.2 Electronic PDF Forms

```
\begin{document}
\begin{Form}
\TextField{Name}\qquad \TextField{Date}

\ChoiceMenu[combo]{Gender}{Male, Female}

\ChoiceMenu[radio]{Project}{cookies, cakes, chocolates, raygun}
```

Which ice cream flavours do you like?

```
\CheckBox{vanilla}
\CheckBox{mint}
\CheckBox{toffee}
\CheckBox{fudge}
\CheckBox[name=guarana]{guaran\^a}
\CheckBox{strawberry}
\CheckBox{raspberry}
\CheckBox{chilli}
\CheckBox{other}

\end{Form}
```

11.2 Electronic PDF Forms

```
\end{document}
```

↓ Input

How the form elements are rendered depends on your PDF viewer. For example, [Figure 11.5](#) shows this form displayed in Adobe Reader and [Figure 11.6](#) shows the same file displayed in Google Chrome. For me, Google Chrome works best (except when it hangs) as there's no native 64 bit Linux version of Adobe Reader, which means I have to run Adobe Reader on Wine and some of the interactive elements cause it to crash. If you use another operating system, you may find that the Adobe PDF viewers, such as Adobe Reader, produce suitable results.

Unfortunately I can't find any other Linux-based PDF viewers that render this example correctly. [Figure 11.7](#) shows the same PDF file viewed in Okular. This renders most of the interactive elements correctly, but fails on the group of radio buttons. Only the first radio button is correctly rendered as an interactive element. The other radio buttons appear as non-interactive open single quote marks. (These appear to be the decorative open quote mark ‘ from the ZapfDingbats font, `\ding{123}`.) A similar problem occurs with Evince and with the document viewer that comes with TeXworks. Other PDF viewers, such as Sumatra or the Linux version of Foxit ([Figures 11.8](#) and [11.9](#)), don't recognise any of the interactive elements (but Foxit on Windows does show the interactive elements, see [Figure 11.10](#)). Therefore, you will need to take care about your choice of

11.2 Electronic PDF Forms

PDF viewer if you want to create an electronic PDF form (and the PDF viewer for any users of your form).

You may have noticed from the above example that each field's label (such as "Name" or "Project") is placed to the left of the interactive element (or elements, in the case of the radio group). This layout is governed by:

`\LayoutTextField{<label>}{<field>}`

Definition

for text fields,

`\LayoutChoiceField{<label>}{<field>}`

Definition

for choice fields, and

`\LayoutCheckField{<label>}{<field>}`

Definition

for check boxes. These all default to `<label> <field>`. Since the space is a regular breakable space, this allowed a line break to occur between the label "raspberry" and its associated check box (as can be seen, for example, in [Figure 11.5](#)). To prevent this, `\LayoutCheckField` can be redefined to use a non-breakable space:

11.2 Electronic PDF Forms

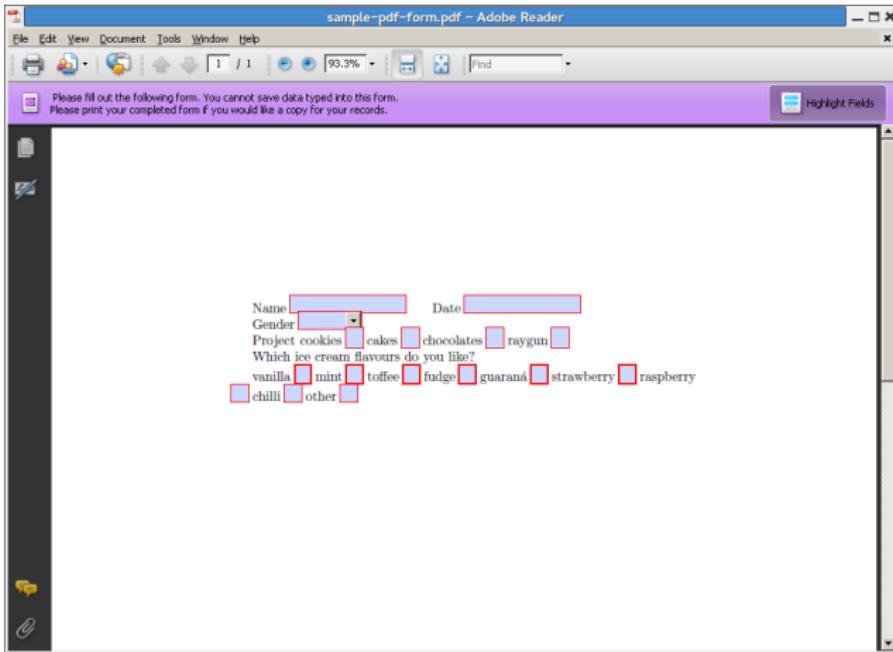


Figure 11.5 Example Interactive PDF Form Viewed in Adobe Reader

11.2 Electronic PDF Forms

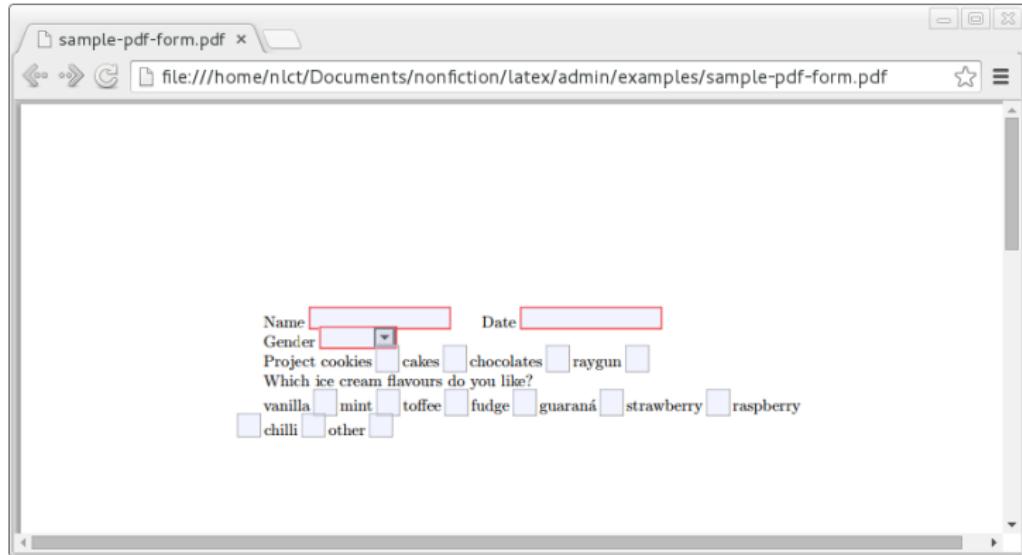


Figure 11.6 Example Interactive PDF Form Viewed in Google Chrome

11.2 Electronic PDF Forms

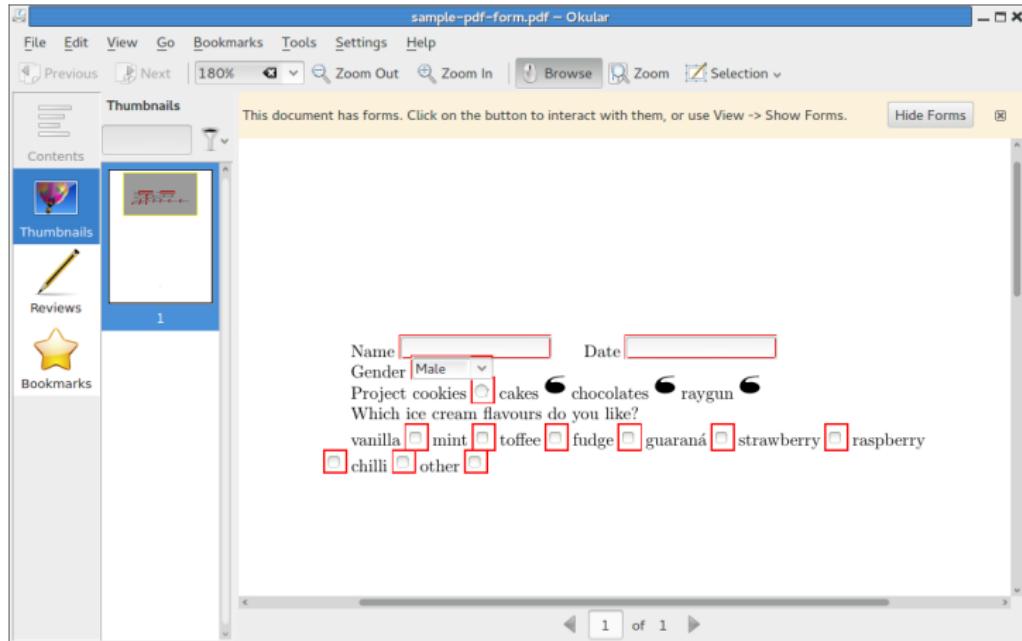


Figure 11.7 Example Interactive PDF Form Viewed in Okular

11.2 Electronic PDF Forms

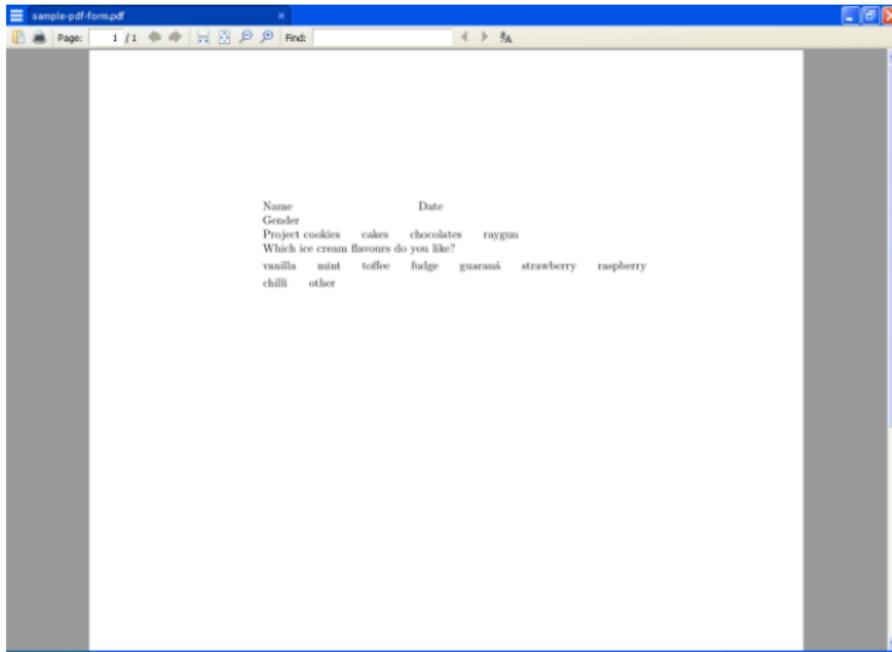


Figure 11.8 Example Interactive PDF Form Viewed in Sumatra

11.2 Electronic PDF Forms

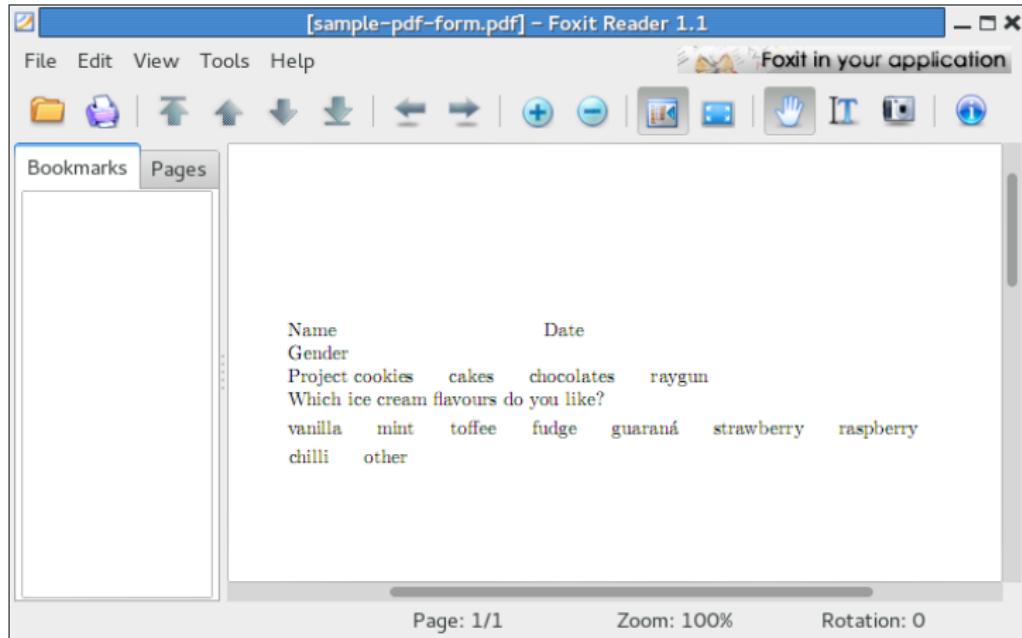


Figure 11.9 Example Interactive PDF Form Viewed in Foxit on Linux

11.2 Electronic PDF Forms

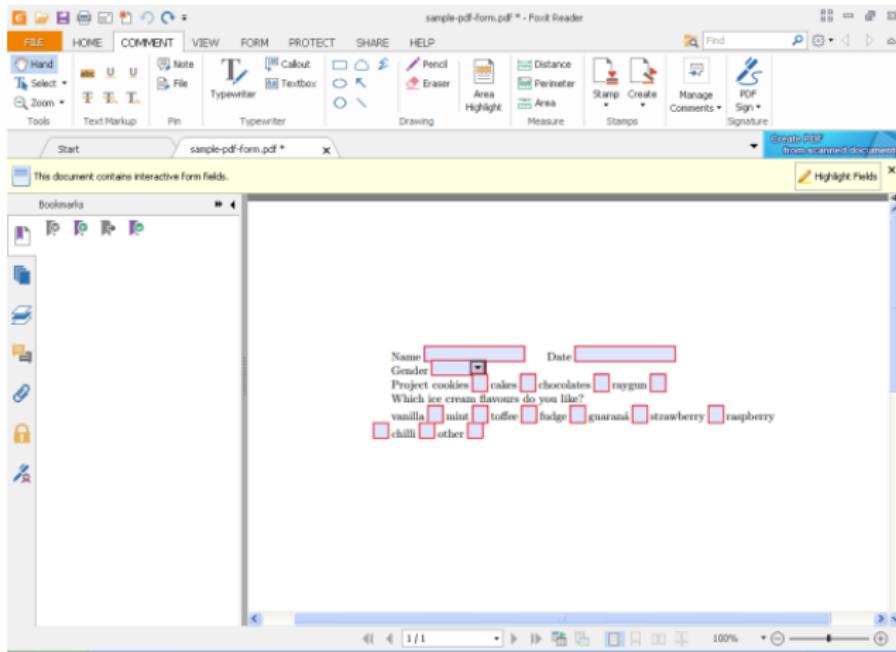


Figure 11.10 Example Interactive PDF Form Viewed in Foxit on Windows

11.2 Electronic PDF Forms

\renewcommand*{\LayoutCheckField}[2]{#1~#2}

Input

Alternatively, if you also want the check box and label swapped round, so that the label is on the right:

\renewcommand*{\LayoutCheckField}[2]{#2~#1}

Input

Note that \LayoutChoiceField just controls the layout of the label for the list of choices and the choice list element or group of elements. In the case of a group of radio buttons, each radio button has a fixed layout with the radio button label first followed by a space and then the radio button. There's no user level macro for changing this layout.

The actual field display is given by:

\MakeRadioField{<width>}{<height>}

Definition

for radio fields;

\MakeCheckField{<width>}{<height>}

Definition

for check boxes;

\MakeTextField{<width>}{<height>}

Definition

for text areas;

11.2 Electronic PDF Forms

\MakeChoiceField{\langle width\rangle}{\langle height\rangle}

Definition

for choice lists. These commands all default to creating a blank area of the given $\langle width \rangle$ and $\langle height \rangle$. (This is how, in [Figure 11.9](#), the area taken up by the interactive elements appears as a blank space even though the fields aren't rendered. The actual rendering of the field is to some extent determined by the PDF viewer.)

The layout of the text on push buttons is determined by

\MakeButtonField{\langle text\rangle}

Definition

This defaults to just $\langle text \rangle$.

The default dimensions are given by the commands:

\DefaultHeightofSubmit

Definition

for the default height of the submit button (14pt);

\DefaultWidthofSubmit

Definition

for the default width of the submit button (2cm);

\DefaultHeightofReset

Definition

for the default height of the reset button (14pt);

11.2 Electronic PDF Forms

\DefaultWidthofReset

Definition

for the default width of the reset button (2cm);

\DefaultHeightofCheckBox

Definition

for the default height of check boxes (\baselineskip);

\DefaultWidthofCheckBox

Definition

for the default width of check boxes (\baselineskip);

\DefaultHeightofChoiceMenu

Definition

for the default height of choice boxes (\baselineskip);

\DefaultWidthofChoiceMenu

Definition

for the default width of choice boxes (\baselineskip);

\DefaultHeightofText

Definition

for the default height of single-lined text fields (\baselineskip);

\DefaultHeightofTextMultiline

Definition

for the default height of multi-lined text fields (4\baselineskip);

11.2 Electronic PDF Forms

\DefaultWidthofText

Definition

for the default width of text fields (3cm). Note that these are all macros not lengths, so you need to use `\renewcommand` to change them. These defaults are used for fields that don't have the height or width options specified (in the optional argument of the field commands, such as `\CheckBox`).

Suppose now I want the check boxes from [Example 58](#) to appear in a tabular layout, so that they appear more like those from [Figure 11.2](#). A first attempt might look something like:

↑ Input

```
\renewcommand*{\LayoutCheckField}[2]{#2 #1}
```

Which ice cream flavours do you like? (Tick all that apply.)

```
\begin{center}
\begin{tabular}{lll}
\CheckBox{vanilla} &
\CheckBox{mint} &
\CheckBox{toffee} \\
\CheckBox{fudge} &
\CheckBox[name=guarana]{guaraná} &
```

11.2 Electronic PDF Forms

```
\CheckBox{strawberry}\\\n\CheckBox{raspberry}&\n\CheckBox{chilli} &\n\CheckBox{other}\n\end{tabular}\n\end{center}
```

↓ Input

However, this produces the form shown in Figure 11.11. The check boxes are far too narrow. Recall from above that the default width of the check boxes is given by `\DefaultWidthofCheckBox`, which is initialised to `\baselineskip`. One of the peculiarities of the `tabular` environment is that it temporarily sets the value of `\baselineskip` to 0 pt. This means that check boxes default to zero width when placed inside a `tabular` environment. Here's a second attempt that changes the defaults to depend on the font size instead:

```
\renewcommand*\LayoutCheckField}[2]{#2 #1}\n\renewcommand*\DefaultWidthofCheckBox}{2ex}\n\renewcommand*\DefaultHeightofCheckBox}{2ex}
```

↑ Input

11.2 Electronic PDF Forms

Which ice cream flavours do you like? (Tick all that apply.)

```
\begin{center}
\begin{tabular}{lll}
\CheckBox{vanilla} &
\CheckBox{mint} &
\CheckBox{toffee} \\
\CheckBox{fudge} &
\CheckBox[name=guarana]{guaraná} &
\CheckBox{strawberry} \\
\CheckBox{raspberry}&
\CheckBox{chilli} &
\CheckBox{other}
\end{tabular}
\end{center}
```

↓ Input

This produces [Figure 11.12](#), which has check boxes with a better width, but the heights are too large causing them to overlap. In fact, they are higher than the specified 2 ex given in the redefinition of `\DefaultHeightofCheckBox`. This seems to be caused by the `tabular` environment stretching the boxes to fill the available height, but this occurs outside of the box created by `\MakeCheckField`.

11.2 Electronic PDF Forms

Here's a third attempt that explicitly sets the width and height within the definition of `\LayoutCheckField` using a `\parbox`:

↑ Input

```
\renewcommand*{\LayoutCheckField}[2]{#2 #1}
\renewcommand*{\DefaultWidthofCheckBox}{2ex}
\renewcommand*{\DefaultHeightofCheckBox}{2ex}
\renewcommand*{\LayoutCheckField}[2]{%
  \parbox[\DefaultHeightofCheckBox]{\DefaultWidthofCheckBox}{#2}%
  #1}
```

Which ice cream flavours do you like? (Tick all that apply.)

```
\begin{center}
\begin{tabular}{lll}
\CheckBox{vanilla} &
\CheckBox{mint} &
\CheckBox{toffee} \\
\CheckBox{fudge} &
\CheckBox[name=guarana]{guaraná} &
\CheckBox{strawberry} \\
\CheckBox{raspberry} &
```

11.2 Electronic PDF Forms

```
\CheckBox{chilli} &  
\CheckBox{other}  
\end{tabular}  
\end{center}
```

↓ Input

This produces the result shown in Figure 11.13, which now has square check boxes. Unfortunately this doesn't take into account the height or width options that may override the default sizes. In this case, that's not an issue as I want all the check boxes to be the same size. If you want larger check boxes in another area of the form, you can localise the effects of the above redefinition of `\LayoutCheckField` by scoping it. For example, by placing it inside the start of the `center` environment before the start of the `tabular` environment:

```
\begin{center}  
\renewcommand*{\LayoutCheckField}[2]{%  
  \parbox[\DefaultHeightofCheckBox]{\DefaultWidthofCheckBox}{#2}  
#1}%  
\begin{tabular}{lll}
```

↑ Input

↓ Input

11.2 Electronic PDF Forms

The other possibility is to use the [internal commands](#) `\Fld@width` and `\Fld@height`, which store the width and height for the check box:

```
\parbox[\Fld@height]{\Fld@width{#2} #1}
```

Input

However, be careful about using internal commands that aren't part of the L^AT_EX kernel as they may change with future versions. Also remember that internal commands must be placed inside a class or package or should be enclosed inside [`\makeatletter... \makeatother`](#).

Which ice cream flavours do you like? (Tick all that apply.)

vanilla	mint	toffee
fudge	guaraná	strawberry
raspberry	chilli	other

Figure 11.11 First Attempt at Laying Out Check Boxes in Rows and Columns

11.2 Electronic PDF Forms

Which ice cream flavours do you like? (Tick all that apply.)

<input type="checkbox"/>	vanilla	<input type="checkbox"/>	mint	<input type="checkbox"/>	toffee
<input type="checkbox"/>	fudge	<input type="checkbox"/>	guaraná	<input type="checkbox"/>	strawberry
<input type="checkbox"/>	raspberry	<input type="checkbox"/>	chilli	<input type="checkbox"/>	other

Figure 11.12 Second Attempt at Laying Out Check Boxes in Rows and Columns

Which ice cream flavours do you like? (Tick all that apply.)

<input type="checkbox"/>	vanilla	<input type="checkbox"/>	mint	<input type="checkbox"/>	toffee
<input type="checkbox"/>	fudge	<input type="checkbox"/>	guaraná	<input type="checkbox"/>	strawberry
<input type="checkbox"/>	raspberry	<input type="checkbox"/>	chilli	<input type="checkbox"/>	other

Figure 11.13 Third Attempt at Laying Out Check Boxes in Rows and Columns

12. CHARTS

Charts and diagrams can be produced in any graphical application that can export the image to a format that \LaTeX can input. However it is also possible to write \LaTeX code to generate the diagram. This has the advantage in that the fonts used in the diagram match those used in the rest of the document, but it's more complicated and can significantly slow the document build time.

This chapter describes \LaTeX packages to generate various charts you may need in your administrative work. If you prefer to use a graphics application to generate a chart you can input the exported image using the `graphicx` package, as described in [Volume 1 \[93, §6\]](#), but make sure, if possible, that you export your image using a vector graphics format (such as PDF or EPS¹) rather than a bitmap (such as PNG or JPEG).

There are many \LaTeX packages available, ranging from general drawing packages, such as `tikz` or `pstricks`, to packages designed for specific types

¹Note that the PDF and EPS file formats also support bitmaps so, if possible, check the settings on whatever application you use to create the image files to see if it uses a vector graphics format. If the image appears fuzzy when you magnify it, then it's most likely a bitmap.

of charts. See, for example, the [diagram topic](#) and sub-topics such as the [diagram-block topic](#) (block diagrams) and [diagram-ctrl topic](#) (control diagrams), as well as the [genchart topic](#) (bar- or pie-charts), [planning topic](#) (timelines and schedules) and [gantt topic](#). There's also the [pgf-tikz topic](#) (for packages that use pgf/tikz) and the [pstricks topic](#) (for packages that use pstricks).

 With the increase in computer graphics over the last couple of decades, there has been a corresponding rise in jazzed-up three-dimensional charts designed to impress the lay person. Such charts can be found from glossy brochures to company annual reports or news programs, but while these images may appear visually appealing, they distort the data and can produce a misleading impression. As a chartered mathematician I can't condone such deception, whether done by design or accident, so I'm not going to show you how to produce fancy effects.

 Be careful if you have large numbers or you may get the "Dimension too large" T_EX error. If you are dealing with very large values (in terms of magnitude), you may be better off using a custom data-handling tool to generate the image rather than trying to use T_EX.

12.1 Flow Charts

12.1 Flow Charts

The [diagram-flow topic](#) lists several packages for flow and similar diagrams, only two of which are available on both MiK_TE_X and T_EX Live, and only one of these is for flow charts, and that's the `flowchart` package, which requires the `makeshape` and `tikz` packages. Alternatively, you can just use the `tikz` package directly.

The `tikz` package [102] has already been briefly introduced in Sections [7.5](#), [10.3](#) and [11.1](#). Drawing a flow chart will also require the `tikz` libraries² `arrows.meta` and `shapes.geometric`, which can be loaded in the preamble using:

```
\usetikzlibrary{arrows.meta}
\usetikzlibrary{shapes.geometric}
```

↑ Input

↓ Input

The `positioning` library is also useful as it provides convenient ways of positioning nodes:

²These are fairly new libraries, so you'll need an up-to-date version of pgf/tikz in order to use them.

12.1 Flow Charts

\use*tikzlibrary{positioning}*

Input

Recall from §7.5, that within the *tikzpicture* environment, you can use

\path[*path options*] (*position*) node[*node options*] (*node name*) {*text*};

Definition

to position a node or you can use the shortcut:

\node[*node options*] (*node name*) {*text*};

Definition

The position can be specified within [*node options*] using *at*=(*position*) or using the *at* keyword:

\node[*node options*] at (*position*) (*node name*) {*text*};

Definition

Alternatively, if you use the positioning library, the node can be placed relative to another node using one of the following *key*=*value* options:

above Place this node above the location specified in the *value*.

below Place this node below the location specified in the *value*.

left Place this node to the left of the location specified in the *value*.

12.1 Flow Charts

- right** Place this node to the right of the location specified in the `\langle value \rangle`.
- above left** Place this node above left of the location specified in the `\langle value \rangle`.
- above right** Place this node above right of the location specified in the `\langle value \rangle`.
- below left** Place this node below left of the location specified in the `\langle value \rangle`.
- below right** Place this node below right of the location specified in the `\langle value \rangle`.

For each of these options, the `\langle value \rangle` part may simply be in the form `\langle shift \rangle` or in the form `\langle shift \rangle of \langle label \rangle`, where `shift` may be a dimension (or an expression that evaluates to a dimension) or a number (in which case the unit is the tikz unit currently in use). If `of \langle label \rangle` is present then the shift is relative to the node identified by `\langle label \rangle`. If the `\langle shift \rangle` part is omitted, the default node distance is used. For further details, and for details of other placement options, see the pgf manual [102].

12.1 Flow Charts

EXAMPLE:

```
\begin{tikzpicture}
\node (start) {Ray-gun doesn't work};
\node[below=of start] (query) {Is it charged?};
\node[right=of query] (recharge) {Recharge battery};
\node[below=of query] (repair) {Repair ray-gun};
\end{tikzpicture}
```

↑ Input

↓ Input

This produces the image shown in [Figure 12.1](#).

At the moment this doesn't look much like a flow chart. The nodes all default to a rectangular shape, but the shape isn't visible unless you use `draw`, for the outline, or `fill`, for the interior, within the `<node options>` specifications. In both cases, you can optionally supply a colour name. Since the `xcolor` package is automatically loaded by `tikz`, you can apply colour mixtures using the `!` specification, such as `red!50` to indicate 50% red (see the `xcolor` documentation [41] for further details).

EXAMPLE:

12.1 Flow Charts

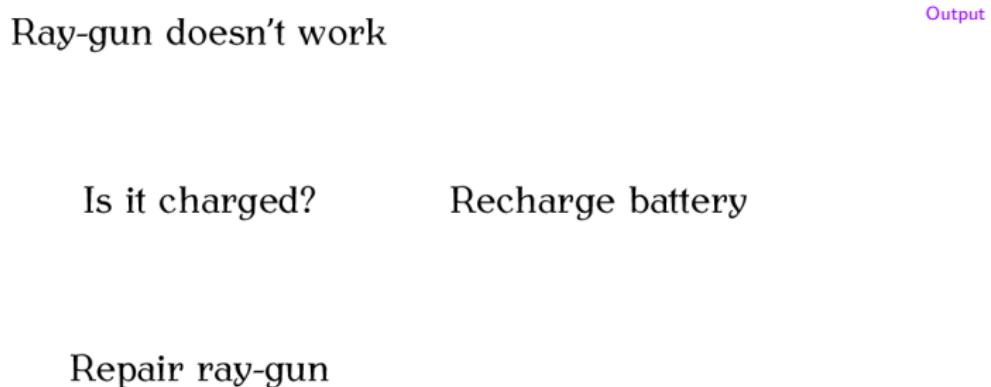


Figure 12.1 Nodes Positioned Relative to Each Other

12.1 Flow Charts

```
\begin{tikzpicture}
\node[draw,fill=red!30] (start) {Ray-gun doesn't work};
\node[draw,fill=yellow,below=of start] (query) {Is it charged?};
\node[draw,fill=green!40,right=of query] (recharge)
{Recharge battery};
\node[draw,fill=green!40,below=of query] (repair)
{Repair ray-gun};
\end{tikzpicture}
```

↑ Input

↓ Input

This produces the image shown in [Figure 12.2](#).

The rectangles can be given round corners using the `rounded corners` option. For example:

```
\node[rounded corners,draw,fill=green!40,below=of query]
(repair) {Repair ray-gun};
```

↑ Input

↓ Input

12.1 Flow Charts

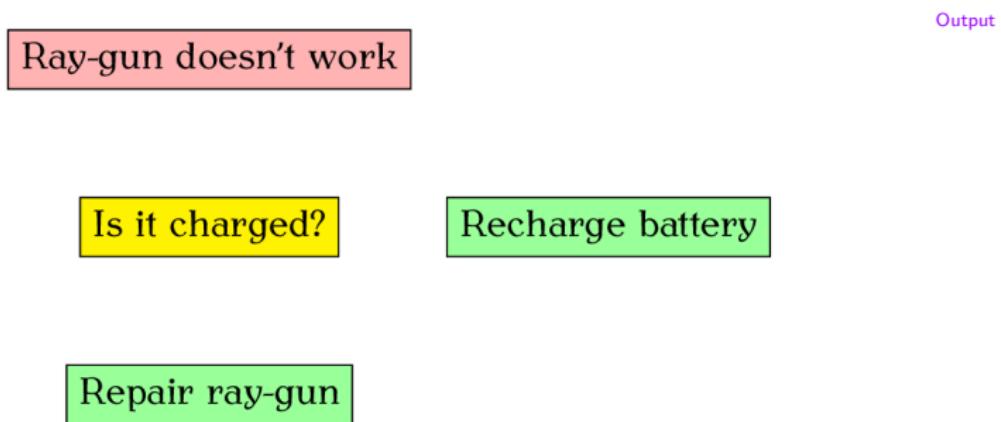


Figure 12.2 Node Shapes Drawn and Filled

12.1 Flow Charts

If you want to change the node shape, there are a number of shapes provided by various tikz libraries. For example, the diamond shape is provided by the `shapes.geometric` library. The shape name is given in the `\node` options. For example:

```
\node[diamond,draw,fill=yellow,below=of start] (query)  
  {Is it charged?};
```

↑ Input

↓ Input

The default aspect ratio of the diamond width and height is 1. You can change this using the `aspect` option. For example:

```
\node[diamond,aspect=2,draw,fill=yellow,below=of start] (query)  
  {Is it charged?};
```

↑ Input

↓ Input

A line can be drawn between two nodes using:

```
\draw[<options>] (<node1 label>) -- (<node2 label>);
```

Definition

For example:

12.1 Flow Charts

```
\draw (start) -- (query);
```

Input

Arrow heads can be added to the start and end of the line using the option `(start arrow)-<end arrow>`, where `(start arrow)` and `(end arrow)` indicate the type of arrow head. The simplest arrow types are given by `<` for an arrow head pointing to the start and `>` for an arrow head pointing to the end. The `(start arrow)` or `(end arrow)` may be omitted if no arrow head is needed at the start or end, respectively.

EXAMPLE:

```
\begin{tikzpicture}
\node[rounded corners,draw,fill=red!30] (start)
  {Ray-gun doesn't work};
\node[diamond,aspect=2,draw,fill=yellow,below=of start] (query)
  {Is it charged?};
\node[draw,fill=green!40,rounded corners,right=of query]
  (recharge) {Recharge battery};
\node[draw,fill=green!40,rounded corners,below=of query]
  (repair) {Repair ray-gun};
% draw in arrows:
\draw[->] (start) -- (query);
```

↑ Input

12.1 Flow Charts

```
\draw[->] (query) -- (recharge);
\draw[->] (query) -- (repair);
\end{tikzpicture}
```

Input

This produces the image shown in [Figure 12.3](#).

A node can be added to a path. For example:

```
\draw[->] (query) -- (recharge) node[midway,above] {No};
```

Input

This places a node (with the text “No”) above and midway along the line between the query and recharge nodes.

EXAMPLE 59. FLOW CHART

This example builds on the above. The `arrows.meta` library is loaded in order to use the `Triangle[]` arrow tip. This can be used by replacing the `>` arrow tip specifier in the optional argument to `\draw`. For example:

```
\draw[-{Triangle[]}]} (start) -- (query);
```

Input

Alternatively, the `>` arrow tip specifier can be set to `Triangle[]` for the given scope. For example:

```
\begin{tikzpicture}[>={Triangle []}]
```

Input

12.1 Flow Charts

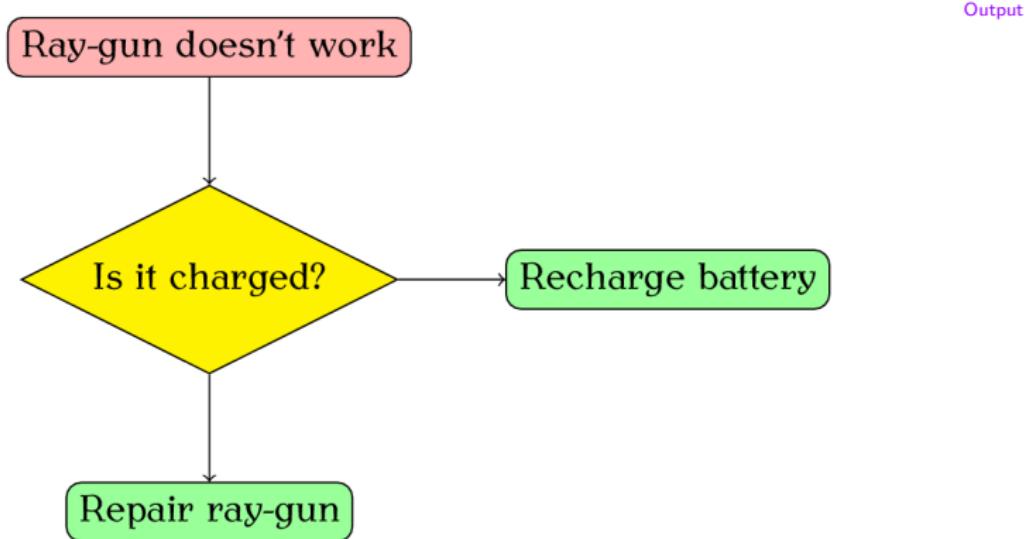


Figure 12.3 Nodes with Connecting Arrows

12.1 Flow Charts

This helps to ensure consistent arrow tips within the picture and means that you only need to edit one line if you decide to change the arrow tips (for example, from `Triangle[]` to `Stealth[]`).

Common node settings can be specified using `every node/.style={<node options>}` within the optional argument of the `tikzpicture` environment (to apply to all nodes within the environment) or the effect can be scoped using the `scope` environment (recall §7.5). This method can be used for the common settings for the recharge and repair nodes.

In addition, a thicker line width is set using the `ultra thick` option.

↑ Input

```
\documentclass{article}

\usepackage{tikz}[2013/12/13] % use at least version 3.0
\usetikzlibrary{arrows.meta}
\usetikzlibrary{shapes.geometric}
\usetikzlibrary{positioning}

\begin{document}

\begin{tikzpicture}[ultra thick,>={Triangle[]}]
```

12.1 Flow Charts

```
\node[rounded corners,draw,fill=red!30] (start)
  {Ray-gun doesn't work};
\node[diamond,aspect=2,draw,fill=yellow,below=of start] (query)
  {Is it charged?};
\begin{scope}[every node/.style={draw,fill=green!40,
rounded corners}]
\node[right=of query] (recharge) {Recharge battery};
\node[below=of query] (repair) {Repair ray-gun};
\end{scope}
\draw[->] (start) -- (query);
\draw[->] (query) -- (recharge) node[midway,above] {No};
\draw[->] (query) -- (repair) node[midway,right] {Yes};
\end{tikzpicture}

\end{document}
```

↓ Input

Note that the last three lines of the `tikzpicture` environment above can also have the arrow tips automatically added through the use of the `scope` environment:

```
\begin{scope}[->]
```

↑ Input

12.2 Pie Charts

```
\draw (start) -- (query);
\draw (query) -- (recharge) node[midway,above] {No};
\draw (query) -- (repair) node[midway,right] {Yes};
\end{scope}
```

[↓ Input](#)

This produces the image shown in Figure 12.4. You can [download](#) or [view](#) this example.



12.2 ■ Pie Charts

At the time of writing, there are three pie chart packages listed on the [genchart topic](#): pgf-pie (which uses pgf/tikz), piechart (shell and AWK scripts to generate pie-charts expressed as pstricks code) and piechartmp (which uses MetaPost). The pgf-pie package is available on MiK_TE_X but is not on T_EX Live.³ The piechart bundle isn't available on either MiK_TE_X or T_EX Live, and only comes with a README file dated 1998. The piechartmp package is available on both MiK_TE_X and T_EX Live, but requires writing MetaPost code which hasn't been covered in this series of books. In addition to these

³The licence is unknown and therefore has to be assumed to be non-free.

12.2 Pie Charts

Output

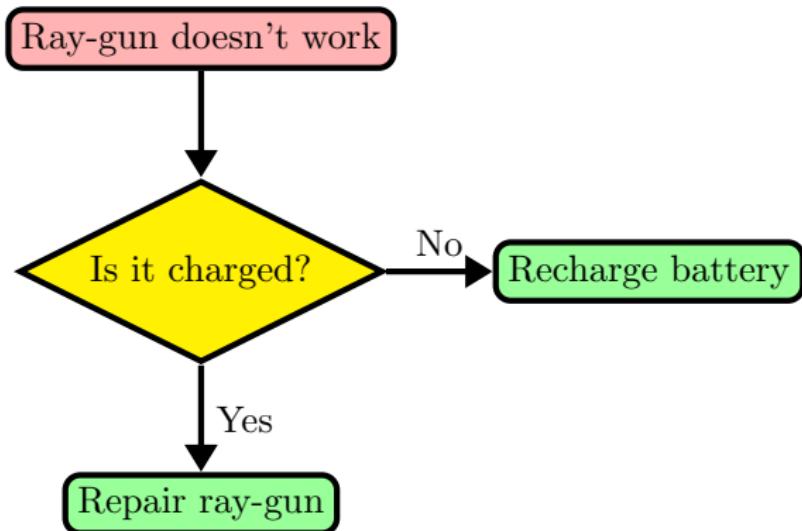


Figure 12.4 An Example Flow Chart

12.2 Pie Charts

packages, the `datatool` bundle comes with the `datapie` package, which can be used to display pie charts from data stored in a `datatool` database.

Since the `datatool` package has already been introduced in this book, §12.2.1 discusses the `datapie` package. If you don't want the additional overhead that comes with the `datatool` package, §12.2.2 discusses the `pgf-pie` package, but TeX Live users will have to install the package manually.

12.2.1 The `datapie` Package

As mentioned above, the `datapie` package is part of the `datatool` bundle [95]. The `datapie` package automatically loads the `datatool` and `tikz` packages. The following package options are provided:

- `color` Use colour (default).
- `gray` Use greyscale.
- `rotateinner` Rotate the inner labels so that they are aligned with the pie chart radial axis.
- `norotateinner` Don't rotate the inner labels (default).

12.2 Pie Charts

rotateouter Rotate the outer labels so that they are aligned with the pie chart radial axis.

norotateouter Don't rotate the outer labels (default).

Once you have loaded the data (see §2.2), the numerical data within the database can be displayed as a pie chart using:

`\DTLpiechart[<condition>]{<settings>}{'<db-name>'}{<assign list>}`

Definition

The optional argument *<condition>* is the same as that for `\DTLforeach`, *<db-name>* is the label uniquely identifying the database, and *<assign list>* is a [comma-separated list](#) of *<cmd>=<col-label>* pairs, the same as the penultimate argument of `\DTLforeach`. The remaining argument *<settings>* is a [key=value list](#). The *variable* key must be present, but the remaining keys may be omitted.

variable The command to use (as specified in *<assign list>*) that contains the data to be used to construct the pie chart.
(Required.)

start The starting angle (degrees) of the first segment. The default is 0.

12.2 Pie Charts

- radius** The radius of the pie chart. The default is 2cm. This sets the length `\DTLradius`.
- innerratio** The distance from the centre of the pie chart to the point where the inner labels are placed is given by this value multiplied by the radius. This must come after `radius`, if the radius also needs to be set. The default is 0.5.
- inneroffset** The distance from the centre of the pie chart to the point where the inner labels are placed. This may be used instead of `innerratio`. If `inneroffset` is omitted, the `innerratio` is used.
- outerratio** The distance from the centre of the pie chart to the point where the outer labels are placed is given by this value multiplied by the radius. This must come after `radius`, if the radius also needs to be set. The default is 1.25.
- outeroffset** The distance from the centre of the pie chart to the point where the outer labels are placed. This may be used instead of `outerratio`. If `outeroffset` is omitted, the `outerratio` is used.

12.2 Pie Charts

- cutawayratio** The distance from the centre of the pie chart to the point of cutaway segments is given by this value multiplied by the ratio. This must come after `radius`, if the radius also needs to be set. The default is 0.2.
- cutawayoffset** The distance from the centre of the pie chart to the point of cutaway segments. This may be used instead of `cutawayratio`. If `cutawayoffset` is omitted, the `cutawayratio` value is used.
- cutaway** The list of cutaway segments. This should be a [comma-separated list](#) of individual numbers, or number ranges (separated by a dash). For example, `cutaway={1,3}` will separate the first and third segments from the rest of the pie chart, whereas `cutaway={1-3}` will separate the first three segments. If omitted, the pie chart will be whole with no cutaway segments.
- innerlabel** The inner label for the segments. The value may contain any of the commands assigned in `\langle assign list \rangle`. The default is the same as the value of the variable `key`.
- outerlabel** The outer label for the segments. The value may contain

12.2 Pie Charts

any of the commands assigned in `<assign list>`. The default is empty.

`rotateinner` This is a [boolean key](#). If true, the inner labels are rotated so that they are aligned with the pie chart radial axis.

`rotateouter` This is a [boolean key](#). If true, the outer labels are rotated so that they are aligned with the pie chart radial axis.

The `datapie` package predefines colours for the first eight segments of the pie chart. If these don't suit your requirements, or if you have more than eight rows of data, you can set the colour for a given segment using:

`\DTLsetpiesegmentcolor{\langle n \rangle}{\langle colour \rangle}`

[Definition](#)

where `\langle n \rangle` is the segment index (starting from 1) and `\langle colour \rangle` is the colour, as used in commands like `\color`.

There are two commands provided that can be used within the inner or outer labels:

`\DTLpievariable`

[Definition](#)

This is set to the value of the `variable` key.

\DTLpiepercent

Definition

This command is set to the percentage value for the current segment. The value is rounded to $\langle n \rangle$ digits, where $\langle n \rangle$ is given by the counter `DTLpierroundvar`.

EXAMPLE 60. AN EXAMPLE PIE CHART (datapie PACKAGE)

Recall the sample `booklist.csv` file. Suppose, for some reason, I want to create a pie chart that displays the price for each book. The database contains 10 rows, so 10 segment colours need to be defined. A pie chart can be drawn as follows:

```
\documentclass{article}

\usepackage[x11names]{xcolor}
\usepackage{datapie}

\DTLloaddb{books}{booklist.csv}

% assign segment colours:

\DTLsetpiesegmentcolor{1}{Aquamarine1}
```

↑ Input

12.2 Pie Charts

```
\DTLsetpiesegmentcolor{2}{Azure2}
\DTLsetpiesegmentcolor{3}{Burlywood3}
\DTLsetpiesegmentcolor{4}{CadetBlue2}
\DTLsetpiesegmentcolor{5}{Chartreuse3}
\DTLsetpiesegmentcolor{6}{Salmon1}
\DTLsetpiesegmentcolor{7}{DeepPink1}
\DTLsetpiesegmentcolor{8}{Goldenrod1}
\DTLsetpiesegmentcolor{9}{Honeydew1}
\DTLsetpiesegmentcolor{10}{Plum3}

\begin{document}

\DTLpiechart{%
variable=\ThePrice,%
innerratio=0.4,%
innerlabel={\$ThePrice},%
rotateinner}%
{books}% database name
{\ThePrice=price}% assignment list

\end{document}
```

↓ Input

12.2 Pie Charts

Output

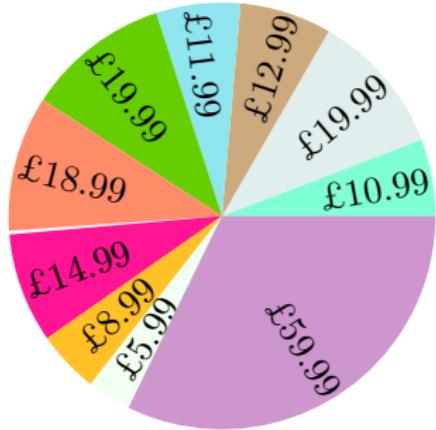


Figure 12.5 An Example Pie Chart (datapie package)

12.2 Pie Charts

This produces the chart shown in [Figure 12.5](#). However, this isn't particularly informative. The book titles could be added as an outer label, but as some of the titles are quite long, this would result in a rather messy chart. Instead, it would be neater to have a legend or key. The current text colour can be switched to the colour of a given segment using:

`\DTLdopiecesegmentcolor{<n>}`

Definition

where $<n>$ is the segment index. This is a declaration that internally calls `\color`. Alternatively, you can use

`\DTLdocurrentpiesegmentcolor`

Definition

which sets the colour for the current segment. This may be used inside a `\DTLforeach` loop:

```
\begin{tabular}{ll}
\DTLforeach*{books}{\TheTitle=title}%
{%
  \DTLiffirstrow{}{\\"}
  \DTLdocurrentpiesegmentcolor\rule{10pt}{10pt} & \TheTitle
}
\end{tabular}
```

↑ Input

12.2 Pie Charts

↓ Input

Recall from Volume 1 [93, §4.7] that the `tabular` environment is a form of box. The pie chart created using `\DTLpiechart` is also a box, so the two can be placed beside each other, however you might need to adjust the vertical alignment. The complete document is as follows:

↑ Input

```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[x11names]{xcolor}
\usepackage{datapie}

\DTLloaddb{books}{booklist.csv}

% assign segment colours:

\DTLsetpiesegmentcolor{1}{Aquamarine1}
\DTLsetpiesegmentcolor{2}{Azure2}
```

12.2 Pie Charts

```
\DTLsetpiesegmentcolor{3}{Burlywood3}
\DTLsetpiesegmentcolor{4}{CadetBlue2}
\DTLsetpiesegmentcolor{5}{Chartreuse3}
\DTLsetpiesegmentcolor{6}{Salmon1}
\DTLsetpiesegmentcolor{7}{DeepPink1}
\DTLsetpiesegmentcolor{8}{Goldenrod1}
\DTLsetpiesegmentcolor{9}{Honeydew1}
\DTLsetpiesegmentcolor{10}{Plum3}

\begin{document}

% pie chart:
\DTLpiechart{%
variable=\ThePrice,%  

innerratio=0.4,%  

innerlabel={\pounds\ThePrice},%  

rotateinner}%
{books}% database name  

{\ThePrice=price}% assignment list  

\qquad% add some horizontal space  

% legend:  

\begin{tabular}[b]{ll}
```

12.2 Pie Charts

```
\DTLforeach*{books}{\TheTitle=title}%
{%
  \DTLiffirstrow{}{\\"}
  \DTLcurrentpiesegmentcolor\rule{10pt}{10pt} &
  \TheTitle
}
\end{tabular}

\end{document}
```

↓ Input

This produces the image shown in [Figure 12.6](#). You can [download](#) or [view](#) this document.

EXERCISE 31. A PIE CHART (datapie PACKAGE)

Recall the sample [orders.csv](#) file. Create a pie chart that displays the values in the quantity column. Alternatively you can use the [orders](#) SQL table.

FOR THE MORE ADVENTUROUS (SQL)

A pie chart showing just the quantity column isn't particularly informative. It would be more interesting to have a pie chart of the total quantities

12.2 Pie Charts

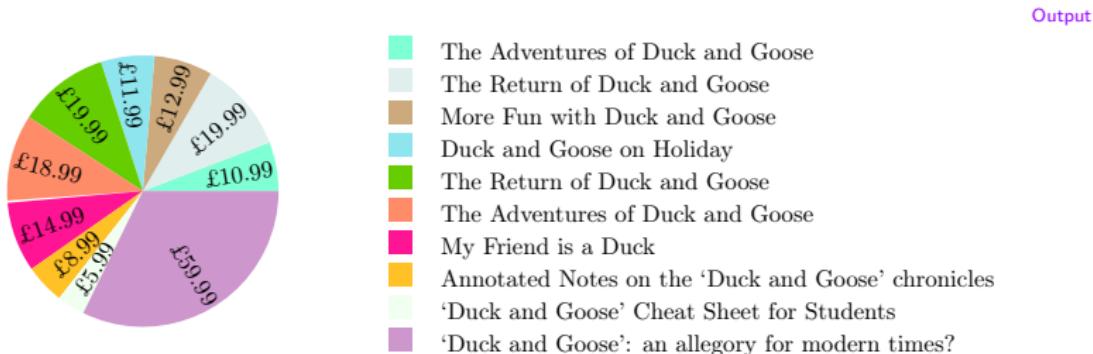


Figure 12.6 An Example Pie Chart with a Legend (datapie package)

12.2 Pie Charts

ordered for each book title (rather than per order) accompanied by the title and format. This requires pulling data from multiple tables, which is far more efficiently done using [SQL](#) than using [TeX](#). The SELECT statement is:

```
SELECT books.title AS booktitle, books.format AS bookformat,  
SUM(orders.quantity) AS total FROM orders, books WHERE  
orders.bookid=books.id GROUP BY orders.bookid
```

[SQL](#)

(If you want the data sorted in descending order of total quantities, you can append ORDER BY total DESC.)

Create a pie chart that shows the order totals for each book with a legend that shows the book title and format.

[\DTLpiechart](#) uses the [tikzpicture](#) environment and there are two hooks available to add additional picture drawing commands to that environment:

[\DTLpieatbegintikz](#)

[Definition](#)

This command is performed before the pie chart is drawn.

[\DTLpieatendtikz](#)

[Definition](#)

This command is performed after the pie chart is drawn.

Use one of these commands to add the legend as a node inside the [tikzpicture](#) environment. Hint: tikz allows radial coordinates specified in the

12.2 Pie Charts

form $(\langle angle \rangle : \langle radius \rangle)$. You can [download](#) or [view](#) the solution to this exercise.

12.2.2 The pgf-pie Package

The pgf-pie package [116] is available on MiK_TE_X, but not on T_EX Live. For most of this book (and for the series, in general) I have chosen packages that are available on both distributions, but as there's so little choice in this topic, this section describes the pgf-pie package. If you are a T_EX Live user, you will need to manually install the package as follows:

1. Download <http://mirror.ctan.org/graphics/pgf/contrib/pgf-pie.zip>
2. Unpack the pgf-pie.zip archive.
3. Copy pgf-pie.sty to somewhere on T_EX's path.
4. Copy the manual pgf-pie-manual.pdf to somewhere on `texdoc`'s path.

For example, on Linux:

12.2 Pie Charts

```
unzip pgf-pie.zip  
mkdir -p ~/texmf/tex/latex/pgf-pie  
cp pgf-pie/pgf-pie.sty ~/texmf/tex/latex/pgf-pie/  
mkdir -p ~/texmf/doc/latex/pgf-pie  
cp pgf-pie/pgf-pie-manual.pdf ~/texmf/doc/latex/pgf-pie/
```

Shell

You can use `kpsewhich` to check that the package has been successfully installed:

```
kpsewhich pgf-pie.sty
```

Shell

This should display the full path to `pgf-pie.sty` if the file is on T_EX's path otherwise it displays nothing.

The `pgf-pie` package provides just one command:

`\pie[<options>]{<list>}`

Definition

which should be placed inside the `tikzpicture` environment. The `<list>` argument should be a `comma-separated list` of values in the form `<number>/<text>` where `<number>` is the value and `<text>` is the outer label for the segment.

The optional argument `<options>` is a `key=value list` where the following keys are available:

12.2 Pie Charts

pos	Sets the centre of the chart to the value $\{\langle x \rangle, \langle y \rangle\}$. The default is the origin.
rotate	Rotates the chart by the given number of degrees.
radius	Sets the radius of the chart. The default is 3.
color	Sets the colours for each segment. The value should be a comma-separated list of colours corresponding to each segment or a single colour, which indicates the colour for the entire chart.
explode	Offset the segments. The value may be a single number, in which case all segments are offset by that amount, or the value may be a comma-separated list of numbers where each value is the offset amount for the corresponding segment.
sum	The sum of all the data. This can be calculated automatically if the auto option is set. The default is 100. If the auto option is off and the actual data sum is less than this value there will be a missing segment in the chart.

12.2 Pie Charts

auto	A boolean key . If true, the sum of the data is calculated automatically.
after number	Indicates the text to place after the number shown in the segment. The default is \%.
before number	Indicates the text to place before the number shown in the segment.
scale font	A boolean key . If true, this scales the font used for the inner label according to the size of the segment, so large segments will have large inner labels and small segments will have small inner labels.
text	Indicates how to position the text (outer label). The value may be one of: label (place the text label outside the segment), pin (as label but also draws a line from the segment arc to the label), inside (place the text label inside the segment above the value) or legend (create a legend). The default is label.

There are some other options as well. See the manual [116] for further details.

12.2 Pie Charts

EXAMPLE 64. A PIE CHART (pgf-pie PACKAGE)

This example has trivial labels A, ..., E and uses the optional argument to change some of the pie chart settings. Note that the pgf-pie package automatically loads tikz.

↑ Input

```
\documentclass{article}

\usepackage{pgf-pie}

\begin{document}

\begin{tikzpicture}
\pie[explode={0,0.5,0,0.75,0},% offset segments 2 and 4
      color={yellow,cyan,green,pink,orange},% segment colour
      text=legend% create a legend
    ]% settings
    {5/A,10/B,15/C,30/D,40/E}% values and labels
\end{tikzpicture}

\end{document}
```

↓ Input

This produces the chart shown in [Figure 12.7](#).

12.3 Bar Charts

There are a number of bar chart packages listed on the [genchart topic](#). Excluding the $\text{\LaTeX}2.09$ entries, at the time of writing they are: bardia g [90], bchart [48] and pst-bar [73]. These three packages are available both on \TeX Live and MiK \TeX . Two of them, bardia g and pst-bar, use pstricks and the other, bchart, uses tikz. Additionally, the datatool bundle [95] provides the package databar, which can be used to display bar charts of data stored in datatool databases.

As mentioned in [§10.4](#), the pstricks package uses PostScript code so it can't be used directly with PDF \LaTeX , so this section will only cover packages that are driver-independent. Therefore [§12.3.1](#) discusses the bchart package and, since this book has already introduced the datatool package, [§12.3.2](#) discusses the databar package, which also uses the tikz package to draw the chart.

12.3 Bar Charts

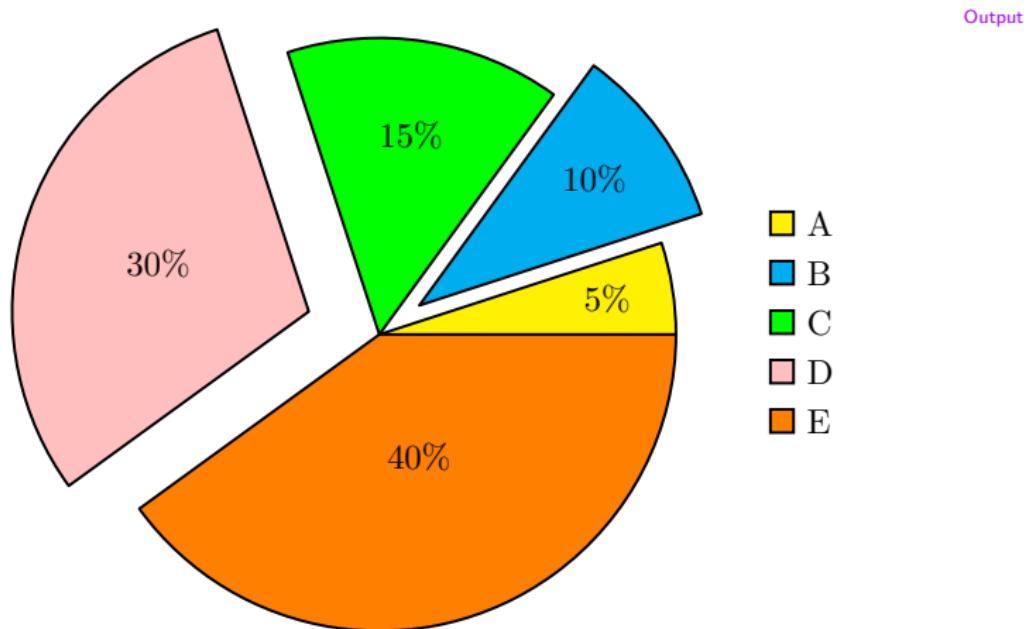


Figure 12.7 An Example Pie Chart (pgf-pie package)

12.3 Bar Charts

12.3.1 The bchart Package

The bchart package [48] provides the `bchart` environment

```
\begin{bchart}[\langle options \rangle]  
  \bar{drawing commands}  
\end{bchart}
```

Definition

The optional argument `\langle options \rangle` is a [key=value list](#) where the following keys are available:

- `max` The maximum x -axis value. (The default is 100.)
- `min` The minimum x -axis value. (The default is 0.)
- `step` The step size along the x -axis.
- `steps` For irregular intervals or if rounding errors cause a problem for the `step` option, this option can be used instead. The value should be a [comma-separated list](#) of intervals along the x -axis.
- `plain` Hides all the tick marks along the x -axis.
- `unit` Specifies a unit to append to all the values displayed on the chart. The default is empty.

12.3 Bar Charts

width Sets the width of the chart. The default is 8 cm. This setting doesn't change the height of the chart.

scale Scales both the width and height, maintaining the aspect ratio. (This doesn't change the text size.)

The font declaration used for the text displayed in the bar chart is given by:

`\bcfontstyle`

Definition

This can be redefined using `\renewcommand` or set to empty to use the document font. The default definition is `\sf`, which is an obsolete font changing command and may cause issues with some classes such as the KOMA-Script classes. I suggest you redefine `\bcfontstyle` to use a modern declaration, such as `\sffamily`. For example:

`\renewcommand*{\bcfontstyle}{\sffamily}`

Input

Within the `bchart` environment, the bars are displayed using:

`\bcbar[<options>]{<number>}`

Definition

where `<number>` is the bar's value. The optional argument `<options>` is a `key=value list` with the following keys:

12.3 Bar Charts

text Sets the text displayed inside the bar (to the right of the *y*-axis).

label Sets the label displayed on the left of the *y*-axis.

color Sets the bar colour.

plain Hides the bar's value, which by default is displayed to the right of the bar.

value Sets the value displayed to the right of the bar. The default is the *(number)* in the mandatory argument of `\bchart`.

You can insert vertical gaps between bars using:

`\bcskip[<options>]{<length>}`

Definition

Within the scope of the `bchart` environment, the standard vertical skips `\smallskip`, `\medskip` and `\bigskip` are redefined in terms of `\bcskip` and may be used to insert small, medium or large gaps. As with `\bcskip`, these three commands also have an optional argument. This option only has one key available: `label`, which specifies a label.

A “free” label is placed to the left of the *y*-axis at the current location using:

12.3 Bar Charts

`\bclabel{<text>}`

Definition

where `<text>` is the label text. This doesn't add any gap or bar to the chart.

The x -axis can be labelled using:

`\bcxlabel{<text>}`

Definition

EXAMPLE:

Here's the data from the pie chart in [Example 61](#) reproduced as a bar chart:

```
\begin{bchart}[max=40,step=5]
\bcbar[label=A,color=yellow]{5}
\bcbar[label=B,color=cyan]{10}
\bcbar[label=C,color=green]{15}
\bcbar[label=D,color=pink]{30}
\bcbar[label=E,color=orange]{40}
\end{bchart}
```

↑ Input

↓ Input

This produces the chart shown in [Figure 12.8](#).

12.3 Bar Charts

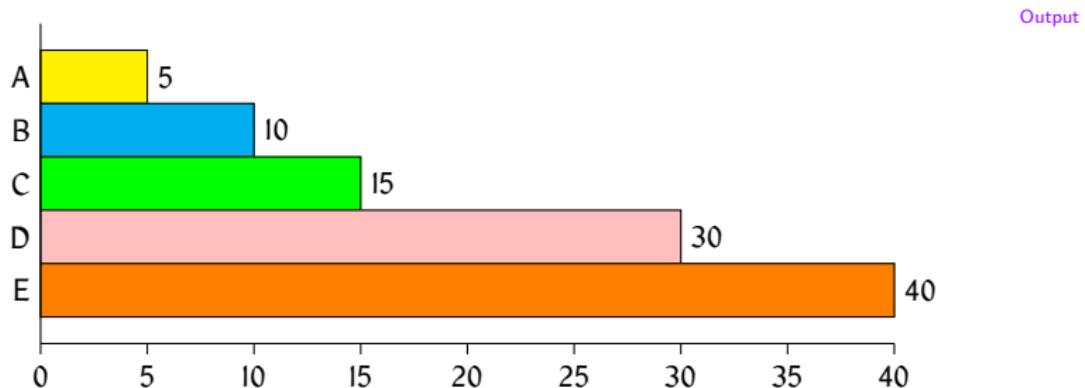


Figure 12.8 A Bar Chart (bchart package)

12.3 Bar Charts

EXERCISE 32. A BAR CHART (bchart PACKAGE)

Reproduce the bar chart shown in Figure 12.9. (The bar colours are the default setting.) Hint: if you actually try to enter values such as 75000 you are likely to get a “Dimension too large” error, so enter the numbers as one thousandth of the actual value and set the unit key so that the numbers are displayed as shown.

You can [download](#) or [view](#) the solution to this exercise.

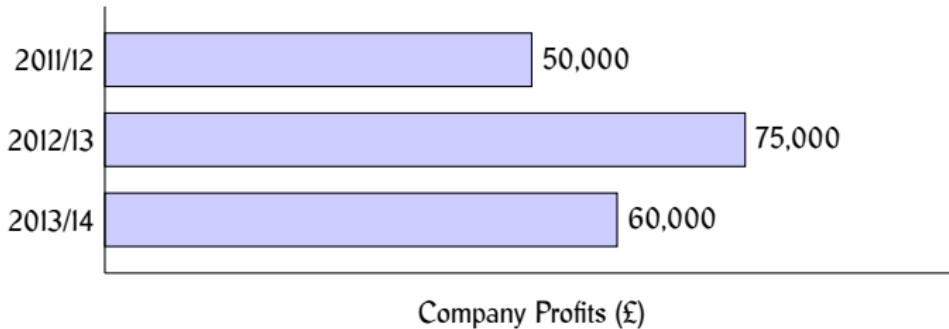


Figure 12.9 Bar Chart (bchart package) Exercise

12.3 Bar Charts

12.3.2 The databar Package

The databar package is part of the datatool bundle [95] and can be used to generate bar charts from data stored in a datatool database. The bar charts may be either vertical or horizontal. The default is vertical. The databar package has the following package options:

- color Create coloured bar charts (default).
- gray Create grey scale bar charts.
- vertical Create vertical bar charts (default). The x -axis is the horizontal axis and the y -axis is the vertical axis.
- horizontal Create horizontal bar charts. The x -axis is the vertical axis and the y -axis is the horizontal axis.

There are two commands provided to generate a bar chart:

`\DTLbarchart[<condition>]{<settings>}{'<db-name>'}{<assign list>}`

Definition

and

12.3 Bar Charts

`\DTLmultibarchart[<condition>]{<settings>}{'<db-name>'}{<assign list>}`

Definition

The former generates a bar chart from a single column of data and the latter generates a bar chart with groups of bars representing multiple columns of data. The `<condition>`, `<db-name>` and `<assign list>` are the same as for `\DTLforeach`. The `<settings>` argument is a `key=value` list where the following keys are available:

variable	This specifies the control sequence (which must be set in <code><assign list></code>) that contains the value used to construct the bar chart. This key is required for <code>\DTLbarchart</code> and is unavailable for <code>\DTLmultibarchart</code> .
variables	This specifies a <code>comma-separated list</code> of control sequences (which must all be set in <code><assign list></code>) that contain the values used to construct the bar chart. This key is required for <code>\DTLmultibarchart</code> and is unavailable for <code>\DTLbarchart</code> .
max	The maximum value on the <i>y</i> -axis. (A decimal number.)
length	The overall length of the <i>y</i> -axis. (A dimension.)

12.3 Bar Charts

maxdepth	A zero or negative number (not a dimension) that specifies the maximum depth of the <i>y</i> -axis.
axes	This may take one of the following values: both (show both axes), <i>x</i> (only show the <i>x</i> -axis), <i>y</i> (only show the <i>y</i> -axis) or none (don't show either axes).
barlabel	Sets the lower bar label. When used with <code>\DTLmultibarchart</code> this indicates the group label.
multibarlabels	This value should be a comma-separated list of labels for each bar within a group for <code>\DTLmultibarchart</code> . This key is not available for <code>\DTLbarchart</code> .
upperbarlabel	The upper bar label. This key is not available for <code>\DTLmultibarchart</code> .
uppermultibarlabels	This value should be a comma-separated list of upper labels for each bar within a group for <code>\DTLmultibarchart</code> . This key is not available for <code>\DTLbarchart</code> .

12.3 Bar Charts

yticpoints	A comma-separated list of tick locations for the <i>y</i> -axis. This setting overrides <code>yticgap</code> .
yticgap	Specifies the gap (a number not a dimension) between the <i>y</i> -tick marks.
yticlabels	A comma-separated list of tick labels for the <i>y</i> -axis.
ylabel	The <i>y</i> -axis label.
groupgap	The gap (a number in terms of the width of a bar) between groups when using <code>\DTLmultibarchart</code> . The default is 1, which indicates one bar width. This key is not available for <code>\DTLbarchart</code> .
verticalbars	A boolean key where <code>true</code> indicates a vertical bar chart and <code>false</code> indicates a horizontal bar chart.

In addition to the above settings, you can also change the appearance of the bar chart by changing any of the following before drawing the chart. Remember that you need to use `\setlength` to change the value of a length register. The *y*-tick labels are rounded to $\langle n \rangle$ digits after the decimal point, where $\langle n \rangle$ is given by the counter `DTLbarroundvar`.

12.3 Bar Charts

\DTLbarchartlength

Definition

This is a length register that stores the total length of the *y*-axis. The default is 3in.

\DTLbarwidth

Definition

This is a length register that stores the width of each bar. The default is 1 cm.

\DTLbarlabeloffset

Definition

This is a length register that stores the distance from the *x*-axis to the lower bar label. The default is 10 pt.

\DTLsetbarcolor{\langle n \rangle}{\langle colour \rangle}

Definition

This sets the $\langle n \rangle$ th bar colour to $\langle colour \rangle$. Only the first eight bars have a colour defined by default. You need to use this command if you need more than eight bars or if you want to override the default colours.

\DTLdobarcolor{\langle n \rangle}

Definition

Sets the current text colour to the colour of the $\langle n \rangle$ th bar.

12.3 Bar Charts

\DTLbaroutlinecolor

Definition

This macro expands to the colour of the bar outlines. This defaults to black. Use \renewcommand to change this value.

\DTLbaroutlinewidth

Definition

A length register that stores the line width for the bar outlines. If it is set to 0 pt, the outline is not drawn. The default value is 0 pt.

Both \DTLbarchart and \DTLmultibarchart draw the chart inside a tikzpicture environment. You can redefine the following commands to insert code at the start or end of this environment:

\DTLbaratbegintikz

Definition

for the hook at the start (after the unit vectors are set) and

\DTLbaratendtikz

Definition

for the hook at the end.

There is also a hook for code to apply at every bar:

\DTLeverybarhook

Definition

Within this book you can use \DTLstartpt (the start of the bar), \DTLmidpt (the mid point of the bar) and \DTLendpt (the end of the bar).

12.3 Bar Charts

There are other commands as well that can be redefined to change the appearance of the bar chart. See the `databar` section of the `datatool` manual [95] for further details.

EXAMPLE 62. AN EXAMPLE BAR CHART (`databar PACKAGE`)

The pie chart from Example 60 can be reproduced as a bar chart as shown below. Since some of the book titles are quite long, the title has been placed inside a `\parbox` to prevent the image from becoming overly tall and the bars are made wider by changing the value of `\DTLbarwidth`.

↑ Input

```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[x11names]{xcolor}
\usepackage{databar}

\DTLloaddb{books}{booklist.csv}

\DTLsetbarcolor{1}{Aquamarine1}
```

12.3 Bar Charts

```
\DTLsetbarcolor{2}{Azure2}
\DTLsetbarcolor{3}{Burlywood3}
\DTLsetbarcolor{4}{CadetBlue2}
\DTLsetbarcolor{5}{Chartreuse3}
\DTLsetbarcolor{6}{Salmon1}
\DTLsetbarcolor{7}{DeepPink1}
\DTLsetbarcolor{8}{Goldenrod1}
\DTLsetbarcolor{9}{Honeydew1}
\DTLsetbarcolor{10}{Plum3}

\setlength{\DTLbaroutlinewidth}{1pt}
\setlength{\DTLbarwidth}{1.2cm}

\begin{document}

\setcounter{DTLbarroundvar}{2}

\DTLbarchart
{variable=\ThePrice,% database column
 axes=both,% show both axes
 barlabel=\parbox{4cm}{\raggedright\TheTitle},% bar labels
 upperbarlabel={\$pounds\{ThePrice\}},% upper bar labels
```

12.4 Gantt Charts

```
yticgap=10,% gap between y tick marks
ylabel={Price (\pounds)}% y-axis label
}% settings
{books}% database
{\ThePrice=price,\TheTitle=title}% assignment list

\end{document}
```

↓ Input

This produces the chart shown in [Figure 12.10](#). You can [download](#) or [view](#) this example.



12.4 ■ Gantt Charts

A Gantt chart (named after Henry Gantt) is a form of bar chart used to illustrate the work breakdown structure of a project. At the time of writing, the [gantt topic](#) has four entries, one of which is for ConTeXt. The remaining three are all L^AT_EX packages available on both MiK^TE_X and T_EX Live. The pgfgantt package [88] uses tikz, and the pst-gantt [27] and rtsched [54] packages use pstricks.

12.4 Gantt Charts

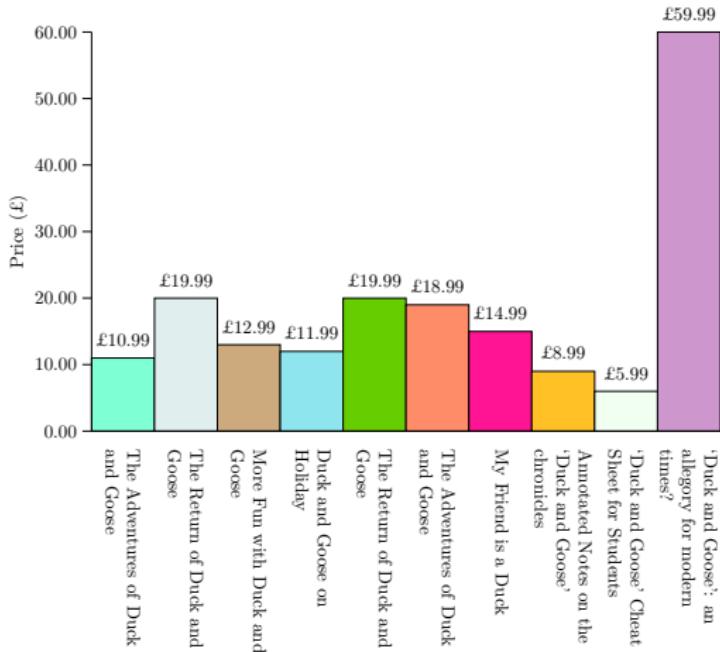


Figure 12.10 A Bar Chart (databar package)

12.4 Gantt Charts

This section discusses the `pgfgantt` package, since it's driver-independent and uses the `pgfcalendar` package, which has already been introduced in §7.2. If you want to use the `pgfgantt` package, make sure you have an up-to-date version of the `pgf` package installed.

The `pgfgantt` package defines the `ganttchart` environment, which can be used to generate a Gantt chart.

```
\begin{ganttchart}[\langle options \rangle]{\langle start tss \rangle}{\langle end tss \rangle}
  \langle entries \rangle
\end{ganttchart}
```

Definition

The optional argument `\langle options \rangle` is a `key=value list`. You can also use these keys in the optional argument of the commands, described below, provided for use within the `ganttchart` environment to apply the setting to just that element. Alternatively you can use

```
\ganttset{\langle options \rangle}
```

Definition

to apply the settings to the current scope.

For the mandatory `ganttchart` environment arguments, `\langle start tss \rangle` is the starting time slot specifier and `\langle end tss \rangle` is the end time slot specifier. The Gantt chart consists of several lines, which may contain title elements or chart elements. Each unit on the chart's `x`-axis represents a time slot.

Note that none of the elements start a new row. Instead you need to

12.4 Gantt Charts

explicitly insert a line break using:

`\ganttnewline[style]`

Definition

where *style* is a key=value list of options permitted by tikz's `\path` optional argument (such as `draw=red`). The default starts a new row without drawing anything.

A time slot specifier (tss) represents a time slot along the horizontal axis. The format used to specify a time slot is determined by the value of the time slot format key (which may be set in *options*). This key may take one of the following values:

- | | |
|--------------------------------|---|
| <code>simple</code> | Each time slot is specified as a positive integer, where 1 indicates the value of the time slot format/start date key (see below), 2 indicates the day after time slot format/start date etc. |
| <code>isodate</code> | Each time slot is specified using the ISO-standard format <code><yyyy>-<mm>-<dd></code> . (The leading zero is optional for the month or day.) |
| <code>isodate-yearmonth</code> | As <code>isodate</code> but without the day <code>-<dd></code> part. The date is assumed to be the first day of the month. |

12.4 Gantt Charts

little-endian	Each time slot is specified in $\langle dd \rangle\text{-}\langle mm \rangle\text{-}\langle year \rangle$ format. (You may also use a slash (/) or period (.) instead of a hyphen (-) for the separator.) A two-digit year will be completed according to the base century setting.
middle-endian	As little-endian but with the $\langle dd \rangle$ and $\langle mm \rangle$ swapped.
big-endian	As little-endian but with the order reversed ($\langle yyyy \rangle$ first and $\langle dd \rangle$ last).

Other keys that may be used with $\langle options \rangle$ include:

time slot format/base century	Sets the century for the auto-completion of two-digit years. The default is 2000.
time slot format/start date	An ISO-standard date $\langle yyyy \rangle\text{-}\langle mm \rangle\text{-}\langle dd \rangle$ denoting the value of the time slot 1 when using the simple format. The default is 2000-01-01, so the time slot 1 indicates 2000-01-01, the time slot 2 indicates 2000-01-02, etc.
canvas/.style	This sets the canvas style. The value should be an option list valid for the

12.4 Gantt Charts

optional argument of a tikz node. (See §7.5 and §12.1.) The default is `{shape=rectangle,draw,fill=white}`, which creates a white canvas with a rectangular frame.

`newline` shortcut

This is a [boolean key](#). If `true` this will allow you to use `\\\` as a shortcut for `\ganttnewline`.

`compress` calendar

This is a [boolean key](#). If `true`, one month corresponds to one time slot, otherwise one day corresponds to one time slot.

There are many other options that govern the chart's formatting, see the pgfgantt manual [88] for further details.

Within the `ganttchart` environment, you can use the commands described below to create chart elements. In each case, `<options>` is as for the `ganttchart` environment.

`\ganttttitle[<options>]{<text>}{<n>}`

Definition

Draws a single title element that covers `<n>` time slots, with the given `<text>`.

12.4 Gantt Charts

`\ganttttitlelist[⟨options⟩]{⟨list⟩}{⟨n⟩}`

Definition

This iterates over `⟨list⟩` and draws a title element that spans `⟨n⟩` time slots for each item in the list. The title text is given by the current list element. The list should be in the format accepted by `\foreach` (see §2.7.2).

`\ganttttitlecalendar[⟨options⟩]{⟨calendar lines⟩}`

Definition

This prints a title calendar that spans the whole chart. The starred form

`\ganttttitlecalendar*[⟨options⟩]{⟨start tss⟩}{⟨end tss⟩}{⟨calendar lines⟩}`

Definition

spans from `⟨start tss⟩` to `⟨end tss⟩`. In both cases, `⟨calendar lines⟩` is a **comma-separated list** of line types:

`year` Print the year.

`month` Print the month number. This may optionally be followed by `=⟨format⟩` where `⟨format⟩` may be one of: `name` (the full name) or `shortname` (abbreviated name).

`week` The week. This may optionally be followed by `=⟨number⟩` where `⟨number⟩` is the number of the first week of the calendar.

12.4 Gantt Charts

weekday The week day number (starting from 0 for Monday). This may optionally be followed by `=⟨format⟩` (analogous to `month`).

day The two-digit day of the month.

There are three chart elements, which can be created using the following:

`\ganttbar[⟨options⟩]{⟨text⟩}{⟨start tss⟩}{⟨end tss⟩}`

Definition

This creates a bar indicating the duration of a task or subtask.

`\ganttgroup[⟨options⟩]{⟨text⟩}{⟨start tss⟩}{⟨end tss⟩}`

Definition

This combines several subtasks into a single task. For the above two commands, `⟨start tss⟩` indicates the starting time slot and `⟨end tss⟩` indicates the end time slot for the task or task group.

`\ganttmilestone[⟨options⟩]{⟨text⟩}{⟨tss⟩}`

Definition

This indicates that a milestone has been completed on the time slot given by `⟨tss⟩`.

If you want to have links between each of these elements (that is, lines drawn from the end of one element to the start of the next element) you can use one of the following commands, analogous to the above three commands.

12.4 Gantt Charts

`\ganttlinkedbar[<options>]{<text>}{<start tss>}{<end tss>}`

Definition

for an individual task or subtask,

`\ganttlinkedgroup[<options>]{<text>}{<start tss>}{<end tss>}`

Definition

for a group, or

`\ganttlinkedmilestone[<options>]{<text>}{<tss>}`

Definition

for a milestone.

EXAMPLE 63. A GANTT CHART

This example is for a Gantt chart that spans a whole year. In this case a day as the time slot would produce a chart that's far too wide, so I've used the `compress calendar` option and I set the `x unit` to 8mm. The pgfgantt manual recommends an `x/y` ratio of approximately 1 : 2 so I've also set both `y unit title` and `y unit chart` to 16mm. The newline shortcut option allows me to use `\\" instead of \ganttnewline`.

```
\documentclass{article}  
  
\usepackage{pgfgantt}
```

↑ Input

12.4 Gantt Charts

```
\begin{document}

\begin{ganttchart}
[time slot format=isodate,%
newline shortcut,%
x unit=8mm,%
y unit title=16mm,%
y unit chart=16mm,%
compress calendar%
]%
options
{2014-1-1}% start time slot
{2014-12-31}% end time slot
\ganttttitlecalendar{year,month=shortname} \\
\ganttgroup{Ray Gun Project}{2014-1-1}{2014-06-30} \\
\ganttbar{Design}{2014-1-1}{2014-04-15} \\
\ganttbar{Testing}{2014-04-01}{2014-06-30} \\
\ganttmilestone{Prototype Ready}{2014-07-01} \\
\ganttgroup{Telepathic Cakes}{2014-5-10}{2014-12-31} \\
\ganttbar{Development}{2014-5-10}{2014-12-31}
\end{ganttchart}
```

```
\end{document}
```

↓ Input

This produces the chart shown in [Figure 12.11](#). You can [download](#) or [view](#) this example.



12.5 ■ Plots

If you have a large amount of data, you may want to consider using a mathematical tool (such as Matlab, Octave or GnuPlot) to generate your graphs as image files. If you use \TeX , you may have excessively long document build times. Don't expect \TeX to be able to compute logarithms or exponentials with the speed or precision of a custom-built numerical computing engine. Additionally, some packages can't parse scientific notation.

There are, however, some packages that use PostScript rather than \TeX to perform the calculations (in which case you can't directly use PDFL \TeX) and some of them rely on \TeX 's [shell escape](#) to call an application, such as GnuPlot, to generate the drawing code (in which case you need the [shell escape](#) enabled). If you have applications such as GnuPlot or Matlab installed or if you are happy to use the `latex+dvips+ps2pdf` route to generate your PDF files, there are a number of useful packages listed on the

12.5 Plots

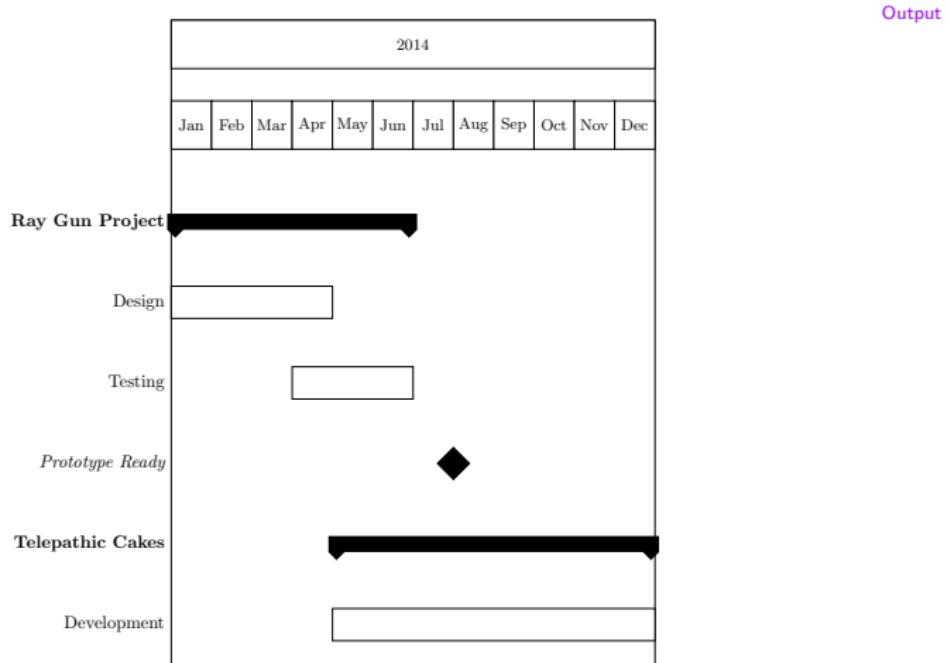


Figure 12.11 A Gantt Chart

12.5 Plots

graphics-plot topic. In addition, pgf/tikz comes with an impressive mathematical engine provided by the pgfmath package. If you want to annotate your plot with text that matches your document format and have the patience to wait for your document to compile, you may be surprised by the images that can be produced using some of the packages available on [CTAN](#).

Since the aim of this book is to be as useful to as many readers as possible, I'm again going to choose device-independent options that don't require any additional applications. (This is, after all, a book on administrative work not high performance computing or advanced mathematics.) As with pie charts and bar charts, the datatool package also provides a package (dataplot) for drawing plots from data stored in a database. However, if you don't intend using that data anywhere else in the document, I suggest you use a more flexible package such as pgfplots, which is described below. The other advantage of pgfplots over dataplot is that pgfplots can parse scientific notation and can cope with larger values. The remainder of this section discusses the pgfplots package.

The pgfplots package [26] uses pgf/tikz so make sure you have an up-to-date version of the pgf package installed. The pgfplots manual [26] is 500 pages long at the time of writing, so this section is a very brief introduction. See the user manual for further details.

Options can be set using

`\pgfplotsset{\langle options \rangle}`

Definition

12.5 Plots

where `<options>` is a [key=value list](#) of options. The pgfplots manual recommends setting the `compat` key to benefit from recent features and to avoid possible changes if you recompile your document at a later date with a newer version. The log file will provide a suggested value.

Other common options include:

<code>width</code>	Sets the width of the final picture. An empty values indicates the default width or rescale in proportion to the height (so that the aspect ratio is maintained if you have set the height).
<code>height</code>	As above but for the height.
<code>scale only axis</code>	This is a boolean key . If true, the above <code>width</code> or <code>height</code> settings apply to the size of the axes rather than the overall picture size (which may include axis labels or tick marks).
<code>xlabel</code>	The x -axis label.
<code>ylabel</code>	The y -axis label.
<code>title</code>	The plot title.

12.5 Plots

`major tick length` The length of major tick marks.

`minor tick length` The length of minor tick marks.

`tick align` The value may be one of: `inside` (default), `outside` or `center`. This indicates the location of the tick marks relative to the axis line.

There are many different types of plots and axes, but this introduction will just consider normal two-dimensional Cartesian plots, which can be produced using the `axis` environment:

```
\begin{axis}[options]
<plot commands>
\end{axis}
```

Definition

This should be placed inside a `tikzpicture` environment. The contents of the `axis` environment may contain one or more instance of

```
\addplot[<plot options>] <plot specs>;
```

Definition

There are a number of different ways of specifying the plot, such as:

```
\addplot[<plot options>] coordinates {<co-ordinate list>} <trailing
path specs>;
```

Definition

12.5 Plots

where $\langle\text{co-ordinate list}\rangle$ is a space-separated list of co-ordinates. A standard Cartesian co-ordinate can be specified in the form $(\langle x \rangle, \langle y \rangle)$, for example, $(0.5, 3.1)$. The $\langle\text{trailing path specs}\rangle$ are optional and are as the path specifications used by tikz's `\path` and `\draw` commands.

```
\addplot[plot options] table [column selection] {file or inline table} trailing path specs;
```

Definition

where $\langle\text{file or inline table}\rangle$ is either a filename or the data arranged in the tabulated form of a space-separated data file. The $\langle\text{column selection}\rangle$ may contain the keys $x=\langle\text{column key}\rangle$, to indicate the column containing the x values, and $y=\langle\text{column key}\rangle$, to indicate the column containing the y values. The table data should include a header row at the start that may be used to reference the columns. If there is no header row, you can use `header=false` within $\langle\text{column selection}\rangle$ and use `x index=<index>` and `y index=<index>` instead of `x` and `y` to reference the required columns. The $\langle\text{index}\rangle$ should be an integer starting from 0 (the first column).

There are other `\addplot` specifications for evaluating expressions or using TeX's `shell escape` mechanism. See the pgfplots manual [26] for further details.

The $\langle\text{plot options}\rangle$ can be any of the path drawing options that are accepted by `\path` (or `\draw`), such as:

`mark` Sets the marker style. There are three standard markers: *

12.5 Plots

(filled circle), x (cross) and + (plus). Additional markers are available with tikz's `plotmarks` library (which can be loaded using `\usetikzlibrary`).

no markers Overrides any `mark` value.

mark repeat The value of this key should be an integer $\langle n \rangle$, to indicate that only every $\langle n \rangle$ th mark should be drawn.

mark size The size of the markers.

dashed Dashed line style.

dotted Dotted line style.

dashdotted Dash-dot line style.

thin Thin line width.

thick Thick line width.

color The value should a colour name used for stroking and filling.
You can omit the `color=` and just write the colour name.

See the pgf manual [102] for further details.

12.5 Plots

EXAMPLE 64. A SAMPLE PLOT (pgfplots PACKAGE)

Suppose instead of the bar chart in [Exercise 32](#) I want a graph of company profits where the x -axis represents the year and the y -axis represents the profit. I could have a file containing that data called, say, `profits.dat` that contains:

```
year profit
2010 52000
2011 50000
2012 75000
2013 60000
```

In which case, I can plot the data using:

```
\addplot table {profits.dat};
```

[Input](#)

or I can just have the data inline as in the example document below:

```
\documentclass{article}
\usepackage{pgfplots}
\begin{document}
```

[↑ Input](#)

12.5 Plots

```
\begin{tikzpicture}
\begin{axis}
\addplot table
{
    year profit
    2010 52000
    2011 50000
    2012 75000
    2013 60000
};
\end{axis}
\end{tikzpicture}
\end{document}
```

↓ Input

This produces the following message in the transcript:

```
Package pgfplots Warning: running in backwards compatibility mode
(unsuitable tick labels; missing features). Consider writing
\pgfplotsset{compat=1.11} into your preamble.
```

So I need to add

12.5 Plots

\pgfplotsset{compat=1.11}

Input

to the preamble. This produces the graph shown in [Figure 12.12](#), which doesn't look right. The x -axis is far too cramped and doesn't need so many tick marks.

The positions of the x -axis tick marks can be changed using the `xtick` key. This may be set to `\empty` (generate automatically), the keyword `data` (use the co-ordinates provided by the first plot), or a [comma-separated list](#) of co-ordinates. (There are also analogous `ytick` and `ztick` keys.)

The tick labels can be changed using various keys. First is the `xticklabels` key which takes a [comma-separated list](#) of labels, where each label corresponds to a tick position. Alternatively, you can use `xticklabel` where the value is the code to produce the label. You can access information about the current tick using the following:

\tick

Definition

This is the current element of the tick option.

\ticknum

Definition

This command is only valid if the `x tick label as interval` option has been set (or `y tick label as interval` for the y -axis ticks) in which case it will contain the position of the next tick position.

12.5 Plots

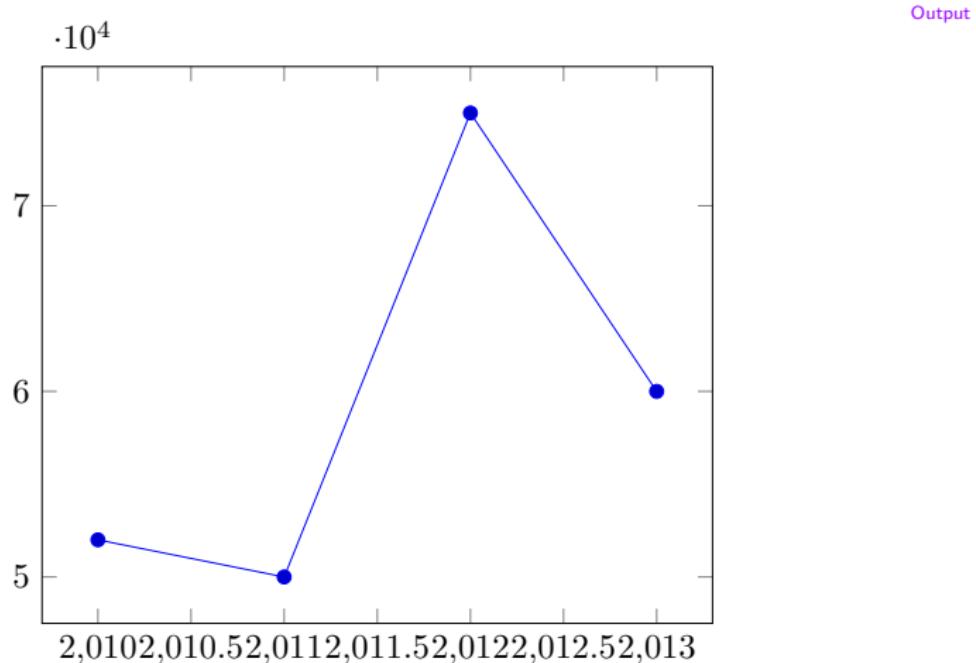


Figure 12.12 A Plot (First Attempt)

12.5 Plots

The default label format is `\pgfmathprintnumber{\tick}`, which uses the number pretty-printing command provided by the `pgfmath` package. This is why the x -tick labels in Figure 12.12 have commas in them (for example, 2,010 rather than 2010). The comma can be removed by first using:

```
\pgfkeys{/pgf/number format/set thousands separator={}}
```

Input

Alternatively, if you just want to change the style for a particular axis:

```
\pgfplotsset{x tick label style={  
    /pgf/number format/set thousands separator={}}  
}
```

↑ Input

↓ Input

The modified document is shown below. I've also added a plot title using `title` and axes labels using `xlabel` and `ylabel`:

```
\documentclass{article}  
  
\usepackage{pgfplots}
```

↑ Input

12.5 Plots

```
\pgfplotsset{compat=1.11}

\begin{document}

\begin{tikzpicture}
\begin{axis}[xtick=data,% get x tick marks from data
 xlabel=Year, ylabel={Profits (\pounds)},% axes labels
 x tick label style=% change x tick label style
 /pgf/number format/set thousands separator={}%
 },
 title={Profits Since 2010},% plot title
 tick align=outside% ticks on the outside
]
\addplot table
{
    year profit
    2010 52000
    2011 50000
    2012 75000
    2013 60000
};

};
```

12.5 Plots

```
\end{axis}  
\end{tikzpicture}  
  
\end{document}
```

↓ Input

This produces the plot shown in [Figure 12.13](#), which is much less cramped.

In order to save space, the y -tick labels have been scaled by 10^4 but since, in this case, the y axis represents profits it would look better if this scaling wasn't applied. There are a number of different ways of changing the scaling (see the manual [26]) but to switch it off for all axes, you just need the option `scaled ticks=false`. It's also possible that you don't like the boxed axes, but this can be changed using the option `axis lines*=left`. These extra options produce the plot shown in [Figure 12.14](#).

You can [download](#) or [view](#) this example where I have additionally loaded the `plotmarks` library and used the optional argument of `\addplot` to set the plot marks to filled diamonds (`mark=diamond*`) with the marker size set to 5 pt (`mark size=5pt`) and a thick cyan line stroke. (Try it for yourself before you download the example, as an additional exercise.)



12.5 Plots

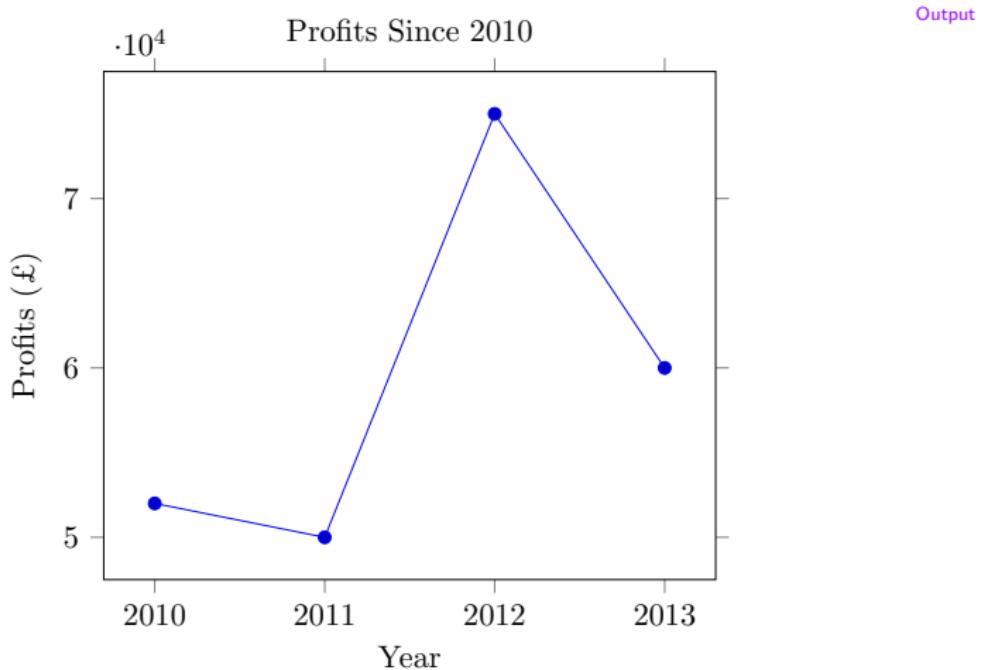


Figure 12.13 A Plot (Second Attempt)

12.5 Plots

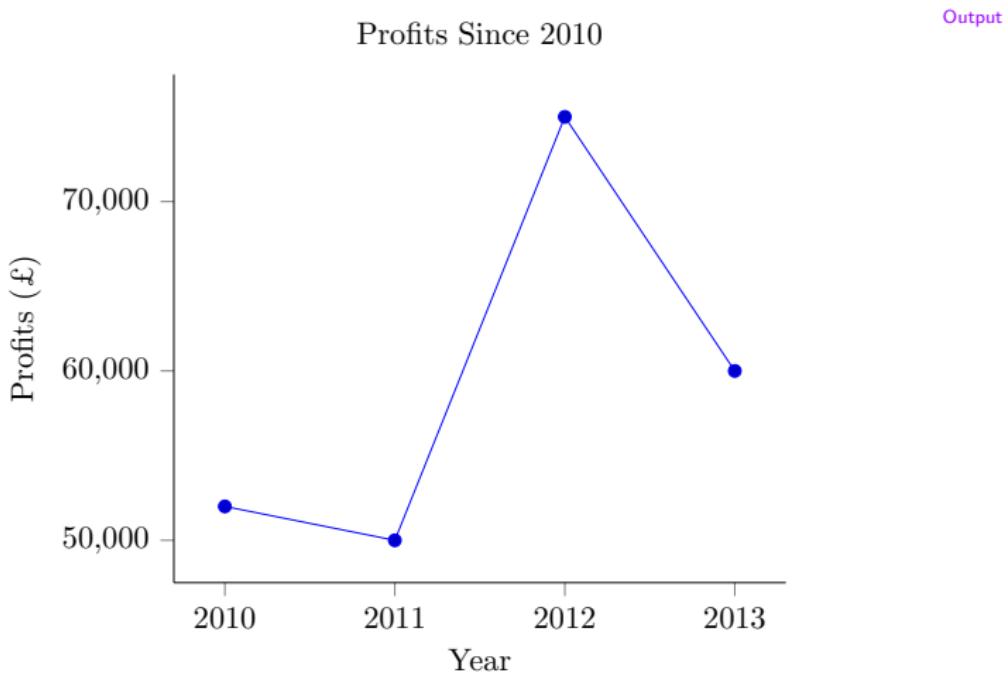


Figure 12.14 A Plot (Final Attempt)

13. COLLABORATING ON DOCUMENTS

There are a number of issues that can occur when multiple authors collaborate on the same document. For example, suppose Fred and Mabel are co-authors of a document, here are some problems they may encounter:

1. If both Fred and Mabel edit their own copies of the source code, how do they merge their edits? If Fred sends Mabel his updated source file, it could overwrite her changes. Similarly if Mabel sends Fred her updated source file, it could overwrite his changes.
2. Fred decides to use package *foo*, which is available on his MiK_TE_X distribution, but Mabel doesn't have *foo* on her T_EX Live distribution.
3. Both Fred and Mabel want to use package *baz* but Mabel has a newer version and is using commands that aren't available in the older version that Fred has installed.
4. Fred and Mabel are working on a sensitive document. If they simply email each other the source code it could be intercepted or if

they transfer it on a portable device, such as a memory stick, it could get stolen.

5. Fred is the principle author and is in charge of writing the first draft. Once he has finished, he sends the document to Mabel who makes minor corrections, but Fred needs to know what modifications she has made.
6. Prof Important Person has decided he's also going to contribute to the document but insists on using his favourite word processor.
7. If Fred or Mabel make an inappropriate change to the document, they may want to roll back to an earlier version of the document to undo the modifications.

The more co-authors on the document, the more likely these problems will occur. The last case is an issue that may also need addressing for a single-authored document. This chapter describes ways of circumventing these problems, but the most convenient solution may not necessarily be the most appropriate solution if you have to take into account factors such as security or available resources or recalcitrant co-authors.

Here are some possible solutions to the above problems:

1. The easiest case is when Fred and Mabel are working on separate sections or chapters. They can then split up the source code and use `\input` (for sections) or `\include` (for complete chapters). Then Fred can just edit the files for his sections and Mabel can just edit the files for her sections. This way they won't be editing the same file at the same time.

In the other case, where Fred and Mabel are both working on the same sections, then the most convenient solutions are a version control system ([§13.2](#)) or using an online L^AT_EX editor ([§13.3](#)).

2. Fred and Mabel agree to only use packages that are available on both MiK^TE_X and T_EX Live, or Mabel manually installs the missing package, or they use an online L^AT_EX editor.
3. Fred and Mabel both update their T_EX distributions before they start working on the document or they use an online L^AT_EX editor.
4. In this case, it depends on the level of security needed. Fred and Mabel may simply be able to use a password-protected version control system. Alternatively, they may need to store the document source code on a portable device that's locked in a safe when the document isn't being edited, and the authors will have to take it in turns to access the device in a non-networked secure environment.

5. Use \LaTeX commands to markup the changes ([§13.1](#)) or use version control.
6. A non- \LaTeX co-author can cause major headaches for a predominantly \LaTeX team of authors. In some cases, gentle persuasion may help. If arguments about the quality of typesetting, the logic of document markup, the convenience of cross-referencing and automatically generated table of contents and similar lists don't have an effect, it might help to point out the security issues associated with Word files, such as the automatic hidden inclusion of personal data and revision information [\[28\]](#).

If persuasion doesn't help, how much of a contribution will they be making to the document? If it's just a matter of a few minor corrections, then it might be possible for them just to mark their changes and for you to then incorporate those changes into the \LaTeX source code. The best way of creating a Word document from the \LaTeX source is to use `tex4ht` to create an OpenDocument Format (.odt) file which can then be converted to Word using an application such as LibreOffice. For example, if the \LaTeX source is in a file called `myDoc.tex` then you can create `myDoc.odt` using:

```
mk4ht oolatex myDoc
```

Shell

7. Use a version control system.

Possible factors that may affect your choice:

Security There are different levels of document sensitivity, for example, information about a surprise event (we're organising Gran's eightieth birthday party, but don't let on); information about a new product that's patent-pending or about to be released with a big media splash; examination papers; information about your country's defence system including launch codes.

Locking up Gran's birthday invitations in a safe within a high-security complex complete with armed guards might be considered overkill. For medium level security, it may be sufficient to have a password-protected version control server located inside your firewall. Servers outside your firewall (including those used by online L^AT_EX editors) may be sufficient for low to medium security documents, but when you use a third-party system you have to put your trust in their security measures. When considering your options, you need to determine the detriment associated with any security breach. (See also §2.3.)

Cost Some of the solutions aren't free. This may impact your choice if you have a low budget.

Operating Systems While some solutions are cross-platform, others might not be. For example, the document build may include running an application or script that only works on a certain operating system. In this case, if possible, it may be better to switch to a cross-platform application (such as one that uses the Java virtual environment) or scripting language (such as Lua, Perl or Python). You may not like the cross-platform alternative, but at least all the authors can build the document.

One or more of the co-authors may have an ancient operating system that doesn't support the latest \TeX distributions, or they may not have administrator privileges required to replace an obsolete \TeX distribution that was installed a decade earlier. The online \LaTeX editors may help with some of these issues, but for security reasons they only allow a limited set of trusted applications (such as `makeindex` or `bibtex`) to run.

To a large extent this issue comes back to cost. The operating system may not be upgraded because the hardware is too old. The IT support may be underfunded and lack the resources for system-wide upgrades. In the case of additional applications to help with the document build, a cross-platform alternative may need to be purchased, if available, or a developer hired to create it.

Recalcitrant Authors Compromises have to be made on any project that involves more than one person. If one or more authors are determined to use a particular set-up or format or application that's incompatible with or inaccessible for the rest of the team then the chances are that the entire project will fail.

The remainder of this chapter is structured as follows:

- §13.1 discusses the changes package to markup changes within the document. This markup can be viewed in the PDF file or can be suppressed for the final version. The changes package can be used with L^AT_EX documents located on shared folders, under version control, or on an online server.
- §13.2 discusses how to access revision information for L^AT_EX documents under version control. Documents under version control can be temporarily locked to prevent conflicting edits, each author can use their favourite text editor to edit the document, and documents can be rolled back to earlier versions. However, all authors must make sure they all have the required class and packages installed.
- §13.3 discusses online L^AT_EX editors. Documents on an online L^AT_EX editor site can be edited by multiple authors, and the authors don't

13.1 Change Markup

need to worry about installing TeX locally, but they have to use the web interface and may not have access to all L^AT_EX tools.

13.1 □ Change Markup

There are many entries listed on the [editorial topic](#) that allow you to mark changes in a document. Some of these are for L^AT_EX2.09, Plain TeX, ConTeXt or LuaTeX, but even discounting these, there are still too many to describe, so I'm just going to describe the changes package [44], which is available on both TeX Live and MiKTeX. The changes package provides a way for the user to manually markup changes, such as additions, deletions or replacements.

The changes package provides an anonymous author that's used by default, but if you want to track the changes according to a particular author, you need to define each tracked author using:

`\definechangesauthor[<options>]{<id>}`

Definition

where $\langle \text{options} \rangle$ is a [key=value list](#) of options and $\langle \text{id} \rangle$ is a label identifying the author. Available options:

name The author's name.

13.1 Change Markup

color The colour to use when marking up this author's changes. (Defaults to black.)

EXAMPLE:

`\definechangesauthor[name={Mabel Canary},color=cyan]{MC}`

Input

To markup changes you can use:

`\added[<options>]{<text>}`

Definition

to indicate that *<text>* has been added,

`\deleted[<options>]{<text>}`

Definition

to indicate that *<text>* has been deleted, and

`\replaced[<options>]{<new text>}{<old text>}`

Definition

to indicate that *<old text>* has been replaced by *<new text>*. Each of these commands has an optional argument that's a **key=value list** where the following keys are available:

id The author's ID (as provided in `\definechangesauthor`).

remark A remark about the change.

13.1 Change Markup

You can create a list or summary of the changes using

`\listofchanges[⟨options⟩]`

Definition

The optional argument is again a [key=value list](#), but there is currently only one key available: `style`, which may have the value `list` or `summary`. The default value is `list`. As with commands such as `\tableofcontents` and `\listoffigures`, this command requires two L^AT_EX runs to produce the list. The default extension for the summary of changes file is `.soc`, but this can be changed using

`\setsocextension{⟨extension⟩}`

Definition

You can suppress the change markup using the `final` package option:

`\usepackage[final]{changes}`

Input

The `draft` option enables the markup. The other package options are `(key)=(value)` options, where the following keys are available:

- `markup=⟨value⟩` sets the markup style. The `⟨value⟩` may be one of:

`default` Colour markup for added text, strike out for deleted text.

`underlined` Added text is underlined, deleted text is struck out.

13.1 Change Markup

bf
Bold for added text and italic for deleted text.

nocolor
As underlined but without colour.

- **addedmarkup=***<value>* and **deletedmarkup=***<value>* set the markup style for added and deleted text. The *<value>* may be one of:

none
No markup.

underline
Underlined text ([example](#)).

uunderline
Double underlined text ([example](#)).

uwave
Wavy underlined text ([example](#)).

dashunderline
Dashed underlined text ([example](#)).

dotunderline
Dotted underlined text ([example](#)).

sout
Struck out text ([example](#)).

xout
Crossed out text ([example](#)).

bf
Bold text.

it
Italic text.

sl
Slanted text.

em
Emphasized text.

13.1 Change Markup

- `authormarkup=<value>` sets the style of the author identification. The `<value>` may be one of:
 - `superscript` Superscript author's name or ID.
 - `subscript` Subscript author's name or ID.
 - `brackets` Author's name or ID in parentheses.
 - `footnote` Author's name or ID in a footnote.
 - `none` No author identification.
- `authormarkupposition=<value>` sets which side of the change to place the author's ID. The `<value>` may be `right` or `left`.
- `authormarkuptext=<value>` sets whether to use the author's ID or name in the change markup. The `<value>` may be either `id` (default) or `name`.

There are also commands to provide custom markup. See the changes documentation [44] for further details.

The changes package automatically loads the `ulem` [2] and `xcolor` packages. You can pass options to these packages using the `ulem` and `xcolor` keys. For example

13.1 Change Markup

```
\usepackage[ulem={normalem,normalbf}]{changes}
```

Input

The changes package also provides

```
\textsubscript{\langle text\rangle}
```

Input

which is analogous to L^AT_EX's `\textsuperscript` kernel command.

 You can't markup floats, such as figures or tables. Paragraph breaks within the markup text can also cause a problem. See the changes documentation [44] for further details.

EXAMPLE 65. RECORDING CHANGES (changes PACKAGE)

This example document has been edited by three authors: Mabel Canary, Fred Canary and Prof Important Person:

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

\usepackage[ulem={normalem,normalbf}]{changes}

\definechangesauthor[name={Mabel Canary},color=violet]{MC}
\definechangesauthor[name={Fred Canary},color=blue]{FC}
\definechangesauthor[name={Prof Important Person},color=teal]{IP}
```

↑ Input

13.1 Change Markup

```
\begin{document}  
\section{About the Lab}
```

The Secret Lab of Experimental Stuff is a
\replaced[id=MC]{top-secret}{sinister} laboratory
whose existence is highly classified so don't tell
anyone about it \added[id=FC]{or we'll get really cross with you}.

The \added[id=IP]{world-renown} University of Somewhere denies all knowledge of the Secret Lab of Experimental Stuff, except on Open Days where members of the public may visit the facility and ask questions as long as they consent to a memory wipe when they leave. The memory wipe is \deleted[id=MC,remark={no it isn't}]{completely} harmless and your memory of the visit will be replaced by a \added[id=MC]{pleasant} recollection of spending the day feeding the \replaced[id=FC,remark={what geese?}]{ducks}{geese} \added[id=MC,remark={they're the weird-looking ducks}]{and geese} in the nearby pond.

```
\end{document}
```

↓ Input

13.2 Version Control

The changes are colour-coded according to the author who made the edit. Any remarks are added as footnotes, as shown in [Figure 13.1](#). You can [download](#) or [view](#) this example.

13.2 ■ Version Control

Version control (also known as revision or source control) is a system that tracks changes to documents and source code. This makes it possible to revert to an earlier version or restore files that are accidentally overwritten or deleted. Typically, the system requires a location where the version control database (repository) is stored (usually password-protected). The authors working on the document pull the latest version of the source code from the database, which creates their own local version of the document files. They can then modify these local files, commit their changes, and push the changed files back to the database.

[FAQ: Version control using RCS, CVS or the like]

The version control system tracks these changes, who made the changes and when they were made. In the event that two or more authors have made conflicting changes to the same piece of text, the version control system will flag the conflict, and the authors need to decide whether to reject their change or overwrite the other author's change.

1 About the Lab

The Secret Lab of Experimental Stuff is a ~~top-secret~~^{sinister^{MC} laboratory whose existence is highly classified so don't tell anyone about it ~~or we'll get really cross with you^{FC}.~~}

The ~~world-renown^{IP}~~ University of Somewhere denies all knowledge of the Secret Lab of Experimental Stuff, except on Open Days where members of the public may visit the facility and ask questions as long as they consent to a memory wipe when they leave. The memory wipe is ~~completely^{MC1}~~ harmless and your memory of the visit will be replaced by a ~~pleasant^{MC}~~ recollection of spending the day feeding the ~~ducks~~^{geese^{FC2} and ~~geese^{MC3}~~ in the nearby pond.}



¹MC: no it isn't

²FC: what geese?

³MC: they're the weird-looking ducks

Figure 13.1 Recording Changes

13.2 Version Control

For example, if Mabel edits section 1 while Fred edits section 2, when they commit their changes there won't be a conflict. The version control system is able to merge their changes so that the source code on the database contains both the new section 1 and the new section 2. Fred and Mabel can then pull this file to update their own local copy. If they decide they don't like the changes, they can revert to an earlier version of the document. If they accidentally delete their local copy, they can pull a fresh copy from the database.

There are a number of different version control systems, some of which are open source, such as [CVS](#), [Subversion](#) (SVN) and [Git](#). Once you have decided which system to use, you need to install the software on your computer and find a location for the database.

In the case of a single author who's only interested in using the version control system as a means of backup, then a local directory can be used. For example, I keep my version control database on an external hard drive. The contents of the external hard drive are periodically backed up to a secondary external device, as an additional precaution. This means that if my primary hard drive fails (which it has done) I can pull all my source code from the database on the external hard drive and I only lose the most recent changes that haven't been committed to the repository. If my external hard drive fails (which it has done) I can restore the repository from the secondary external device and commit my local files. In this case

13.2 Version Control

I lose any of the revisions that haven't been backed up, but I still have the latest version of the document.

In the case of multiple authors, a server will be needed. The choice then comes down to a trade-off between document security and budget. There are free options available for public projects, but private projects usually require payment to a third party hosting service, or a secure server within your institution's firewall can be set up (assuming that all the authors are located within the institution). Some hosting services may be supported by advertisements. Bitbucket allows private hosting for up to 5 users under their free plan and GitHub provides private repositories under their paid plans [37].

Although Fred and Mabel can track all the document changes through the version control system, these changes aren't automatically added to the document and can't be viewed from the resulting PDF. This is often desired, but sometimes it's useful to include some of the revision information, particularly when a document is periodically updated and republished. In this latter case, there are some L^AT_EX packages that can do this, listed in the [version-control topic](#).

For Git users, there's the gitinfo2 package [55] (which replaces the now deprecated gitinfo). For Subversion users, there are four packages available: svn [52], svninfo [9], svn-multi [81] and svn-prov [79]. This last package (svn-prov) is designed for package authors. These packages are all available on both

13.2 Version Control

T_EX Live and MiK_TE_X. There's also the vc package [35] that supports Git and Subversion, but this isn't on either T_EX Live or MiK_TE_X and requires awk. For CVS users, there are packages such as rcs [86], rcsinfo [113] or rcs-multi [80], which are available both on T_EX Live and MiK_TE_X.

 Version control systems usually create hidden files or directories. If you need to bundle up your document and send it to, say, a journal or conference proceedings, make sure you exclude all the hidden files. For example:

```
zip -r <name>.zip <directory> -x "*/\.*"
```

Shell

For Windows users, there's a command line zip program available in [GNU On Windows \(GOW\)](#).

EXAMPLE 66. ACCESSING SUBVERSION REVISION INFORMATION

This book and the accompanying example documents and exercise solutions are in a Subversion repository. In order to insert version control information into one of these documents, I need to add a *keyword anchor*. This is just a bit of text in the form

`$<KeywordName>$`

Input

The `<KeywordName>` indicates the information you want to insert and (for Subversion) may be one of [66]:

13.2 Version Control

- Date The last time the file was known to have been changed in the repository.
- Revision The last known revision in which the file was changed in the repository.
- Author The last known user to change this file in the repository.
- HeadURL The full URL to the latest version of the file in the repository.
- Id A compressed combination of the other keywords.
- Header Similar to Id but contains the full URL of the latest revision.

For example:

\$Id\$

[Input](#)

In addition to adding this text to the file under version control (called, for example, `svninfo-sample.tex`) you also need to tell Subversion to perform a keyword substitution whenever the file is committed. This is done using:

13.2 Version Control

```
svn propset svn:keywords "Id" svninfo-sample.tex
```

Shell

Now, after I next commit the file `svninfo-sample.tex`, the keyword anchor `Id` will be changed to the revision information. For example:

```
$Id: svninfo-sample.tex 383 2014-12-12 19:23:03Z nlct $
```

Input

This will cause a problem when I build my document as L^AT_EX will interpret those dollar `$` symbols as the maths shift special character, so the information won't be typeset correctly. However, recall from §2.1.1 that you can use `\def` to define a command with a particular syntax. For example:

```
\def\svnInfo#${#1}{%
  % do something with #1
}
```

↑ Input

↓ Input

Now I can have

```
\svnInfo$Id: svninfo-sample.tex 383 2014-12-12 19:23:03Z nlct $
```

Input

and the `$` symbols are part of the syntax and aren't interpreted as the maths mode shift. The `svninfo` package provides a command called

13.2 Version Control

\svnInfo \$Id\$

Definition

which is defined in a similar manner. Note that the trailing space is part of the syntax. This command parses the Id keyword anchor and gathers the information for later use in the document. By default this information is placed in the page footer.

For example, here's my original document called `svninfo-sample.tex`:

↑ Input

```
\documentclass{article}

\usepackage{svninfo}

\svnInfo $Id$

\author{Nicola Talbot}
\title{A Sample Document}

\begin{document}
\maketitle
```

This is a sample document.

13.2 Version Control

```
\end{document}
```

↓ Input

If I build this document using pdflatex, the resulting PDF file contains the footer

Rev: -revision-, December 15, 2014

1

-sourcefile-

Output

This is because there's currently no version control information in the file, just the keyword anchor \$Id\$. Next I need to add this new file to my Subversion repository:

```
svn add svninfo-sample.tex
```

Shell

(See the Subversion user manual [66] for information on setting up a repository.) Also, I need to indicate that this file contains the Id keyword anchor:

```
svn propset svn:keywords "Id" svninfo-sample.tex
```

Shell

Now I can commit these changes to the repository:

```
svn commit -m "Add svninfo example"
```

Shell

13.2 Version Control

This not only commits the new file `svninfo-sample.tex` to the repository, but also modifies the file so that it now looks like:

```
\documentclass{article}

\usepackage{svninfo}

\svnInfo $Id: svninfo-sample.tex 385 2014-12-15 11:45:51Z nlct $

\author{Nicola Talbot}
\title{A Sample Document}

\begin{document}
\maketitle

This is a sample document.

\end{document}
```

↑ Input

↓ Input

If I rebuild this document, the PDF file is now as shown in [Figure 13.2](#). The footer now contains the revision number and the filename. You can [download](#) or [view](#) this example.

13.2 Version Control

Output

A Sample Document

Nicola Talbot

December 15, 2014

This is a sample document.

Rev. 385, December 15, 2014

1

svninfo-sample.tex

Figure 13.2 Subversion Revision Information

13.2 Version Control

In addition to `\svnInfo`, the `svninfo` package provides a way to gather information from a different keyword, such as `Revision`, using

`\svnKeyword $⟨value⟩$`

Definition

For example:

`\svnKeyword $Author: nlct$`

Input

Remember that you need to include these extra keywords in the `svn:keywords` property to ensure they are also expanded. For example, if you want to access the `Author` and `Date` rather than the `Id`, then you need to set the keywords using

```
svn propset svn:keywords "Date Author" svninfo-sample.tex
```

Shell

(Keywords are case-sensitive.) The footline can be removed using the `nofancy` package option

`\usepackage[nofancy]{svninfo}`

Input

The Subversion information can be accessed using any of the following commands:

13.2 Version Control

\svnInfoFile

Definition

for the name of the source file or --sourcefile-- if unknown;

\svnInfoRevision

Definition

for the revision number of the checked out file or --revision-- if unknown;

\svnInfoDate

Definition

for the date in the form <YYYY>-<MM>-<DD> when the file was checked out or the current date if unknown;

\svnInfoTime

Definition

for the time when the file was checked out or --time-- if unknown;

\svnInfoOwner

Definition

for user name of the file owner or --owner-- if unknown;

\svnInfoYear

Definition

for the year when the file was checked out or the current year if unknown;

13.2 Version Control

\svnInfoMonth

[Definition](#)

for the month (in $\langle MM \rangle$ form) when the file was checked out or the current month if unknown;

\svnInfoDay

[Definition](#)

for the day (in $\langle DD \rangle$ form) when the file was checked out or the current day if unknown;

\svnInfoLongDate

[Definition](#)

for the date in the form of `\today` when the file was checked out or the current date if unknown.

A summary of the Subversion information can be obtained using

\svnid

[Definition](#)

which uses the same format as the `\Id` keyword anchor. In addition to the above commands, the `svninfo` package also provides commands for multi-file documents. Every file of the document must include either `\svnInfo` or `\svnKeyword` with the `Revision` keyword. Two L^AT_EX runs are required.

\svnInfoMinRevision

[Definition](#)

13.2 Version Control

displays the minimum revision number for all the files in the document, and

`\svnInfoMaxRevision`

[Definition](#)

displays the maximum revision number for all the files in the document. (If unknown, the defaults are `--minrevision--` and `--maxrevision--`.) The date from the latest Subversion revision (in the same format as `\today`) is obtained using:

`\svnInfoMaxToday`

[Definition](#)

If you want the URL to the file in the repository, you need the `HeadURL` keyword

`\svnKeyword $HeadURL:$`

[Input](#)

and you can then use:

`\svnInfoHeadURL`

[Definition](#)

to access this information.

13.3 Online L^AT_EX Editors

Online L^AT_EX editors allow you to store your source code on the provider's server and edit it using your web browser. The source code is compiled using the T_EX distribution installed on the server. This means you don't need to worry about co-authors having a different T_EX distribution (or even having a distribution at all). However, it means that you're limited to the classes and packages installed on the server and they may not be the most recent version.

The two most popular online L^AT_EX editors appear to be [Overleaf](#) (formerly WriteL^AT_EX) and [ShareL^AT_EX](#). The information provided here (such as available features and pricing) may have changed since the time of writing this chapter.

ShareL^AT_EX There are four plans: Personal, Student, Collaborator, and Professional. The pricing and available features are listed in [Table 13.1](#). The site supports the following languages: Italian, Japanese, Korean, Portuguese, Czech, Dutch, Chinese (Simplified), Norwegian, English, German, Danish, Russian, French, Swedish and Turkish.

To create a free personal account, just go to <https://www.sharelatex.com/register> and enter your email address and the password you want to use. To create a new project, go to <https://www.sharelatex.com>.

13.3 Online L^AT_EX Editors

Table 13.1 ShareL^AT_EX Plans (as at December 2014)

		Personal	Student	Collaborator	Professional
GBP	per month	Free	£6	£12	£24
£	per year	Free	£60	£144	£288
EUR	per month	Free	€7	€14	€28
€	per year	Free	€70	€168	€336
USD	per month	Free	\$8	\$15	\$30
\$	per year	Free	\$80	\$180	\$360
Maximum Number of Collaborators		1	6	10	Unlimited
Full Document History?		No	Yes	Yes	Yes
Sync to Dropbox?		No	Yes	Yes	Yes
Sync to GitHub?		No	Yes	Yes	Yes

13.3 Online L^AT_EX Editors

com/project and click on the “New Project” button. This will drop-down a list of options, such as a blank project, upload a project, import from GitHub, or create a project from a template. There’s a wide selection of templates:

- Bibliographies: BibT_EX, thebibliography environment, natbib, biblatex, biblatex with split bibliographies, IEEE BibT_EX style, and notes2bib.
- Books: includes Springer styles, Tufte style, classic thesis, Wiley styles, MIT styles and sffms (science fiction and fantasy manuscripts).
- Exams.
- Cover letters: moderncv styles and Illinois University.
- Other: includes business cards, pst-barcode, homework styles, various letter styles, brochures, flyers, posters, business reports, recipe, lab reports, assignment sheets, invoices, grant applications, and proposals.
- CV or Résumé: includes moderncv styles, europecv styles, classic, academic, professional, fancy, curve, curvita.
- Thesis: there are a lot of templates in this category, some are general and some are specific to particular institutes.

13.3 Online *L^AT_EX* Editors

- Presentations: again there are a lot of templates, many of them using various beamer themes.
- Journals: many different journal styles.

Overleaf There are four plans available: Free, Personal+, Pro and Pro+.

The pricing and available features are listed in [Table 13.2](#). There are also tailored solutions for universities, publishers and enterprise. Students get all the features of the Pro account for half price.

When you create a new project with Overleaf, a unique identifier is generated that's used in the link to the project. The files that make up the project are private as long as that link remains private. It's unlikely that anyone will guess the link, but if it does get published or shared then the project can be accessed and edited. It's your responsibility to ensure you don't share the link with anyone who shouldn't access it. (Be careful of the link appearing in your browser history if you use a shared computer.) With the Pro plans you can create protected projects for added security. Overleaf uses [Amazon S3](#) for secure data storage.

Table 13.2 Overleaf Plans (as at December 2014)

		Personal	Personal+	Pro	Pro+
GBP £	per month per year	Free Free	£5 £48	£6.50 £60	£9.50 £90
EUR €	per month per year	Free Free	€6.50 €60	€7.50 €72	€11.50 €108
USD \$	per month per year	Free Free	\$9 \$84	\$10 \$96	\$15 \$144
Storage space		up to 1GB	2GB	10GB	20GB
Files per project		60	120	240	500
Unlimited projects and collaborators		Yes	Yes	Yes	Yes
Save and restore version history		Yes	Yes	Yes	Yes
Tag, filter and clone projects		Yes	Yes	Yes	Yes
Quick save to Dropbox		No	Yes	Yes	Yes

Table 13.2 Overleaf Plans (Continued)

	Personal	Personal+	Pro	Pro+
Integrated spell-check	No	Yes	Yes	Yes
Editor themes	No	Yes	Yes	Yes
LaTeX auto-complete	No	Yes	Yes	Yes
Compare versions and see tracked changes	No	No	Yes	Yes
Access control on protected projects	No	No	Yes	Yes
Priority support	No	No	Yes	Yes
Full project history (coming soon)	No	No	Yes	Yes

13.3 Online L^AT_EX Editors

Table 13.2 Overleaf Plans (Continued)

	Personal	Personal+	Pro	Pro+
PeerJ Free lifetime Publishing Plan	No	No	Basic	Enhanced

To create a free Overleaf account, just go to <https://www.overleaf.com/signup>, fill in your name and email and click on the create new account button. This takes you to the dashboard and sends you an email with a link to confirm your account registration and choose a password.

To create a new project, click on the “create new project” button in the dashboard. This pops up a window where you can select a template for your document. The available templates are divided into categories:

- Basic: blank paper, sample paper, presentation.
- Academic journals: IEEE Transactions, Springer LNCS, IEEE sponsored conferences and symposia, BioMed Central, MDPI, PeerJ, Elsevier, OSA Express journals, IEEE for Computer Science Journals, IOP journals, Optica, F1000Research, Language Science Press, Springer Journals, APS, Public Library

13.3 Online L^AT_EX Editors

of Science, Advances in Optics and Photonics (AOP), Royal Society Open Science, Copernicus Publications.

- Bibliography: IEEE with BibT_EX, biber and biblatex, natbib, Chicago citation style with biblatex.
- Book: Tufte, ePUB/eBook, Language Science Press, fiction.
- Formal letter: includes newlfm, memo, professional formal letter, Aalto School of Business Letter, Carleton letter
- Homework assignment: a selection of templates for homework or lab reports.
- Newsletter: Tufte handout template, memo template, a newsletter template, a flowfram template.
- Poster: a selection of templates including a sciposter and a beamer template.
- Presentation: a selection of templates, most of them seem to use beamer with different themes.
- Project lab reports: a large selection of report-like templates.
- Résumé/CV: a selection of CV templates.
- Thesis: a large selection of thesis templates, many of them for specific universities.

13.3 Online *LATEX* Editors

One thing I noticed when trying out the templates was that once I had created a project by selecting a template, there was no way to permanently delete it. If you delete a project from the project list on the dashboard, the project is moved to trash and after 30 days, the project will be removed from the dashboard, but the project itself isn't deleted. This was a bit annoying as I only created the projects to see what the templates looked like. Only some of the templates actually included information in their description about the packages used within the template.

I tried both sites using Mozilla Firefox (v33.1) and Google Chrome (v39.0.2171.71) running under 64-bit Fedora 19. In addition, to test accessibility support for visually-impaired users, I tried both sites with the text-only browser Lynx (v2.8.8dev.15) and with the text-to-speech Jovie plugin for Konqueror (v4.11.5).

User Interface

Overleaf The interface was mostly fine on Firefox and Chrome. In general on the site, if the window isn't wide enough for the navigation bar, the navigation links are replaced by a button that drops-down a list of links so a horizontal scroll bar is never required by the browser. When editing a project in the source code pane, there were some

13.3 Online *L^AT_EX* Editors

issues with trying to copy selected text in the source code pane to the clipboard. If I used the popup menu's "copy" item, I received a message "This action isn't yet available from the menu, but you can press Ctrl-C instead." This cleared the selection, and I had to reselect the text and use Ctrl-C. (Firefox's Edit→Copy menu item and Chrome's Copy button worked. The issue was just with the popup menu.)

An example document with Overleaf is shown in [Figure 13.3](#). This has a split pane area with the source code on the left and a preview image on the right. The divider bar between the two panes can be moved to make one pane wider than the other. If the source code pane isn't wide enough for all the menu buttons, you can access the remaining buttons through the "More" drop-down menu.

The preview is automatically updated whenever the file is modified, which I found distracting, but the automatic compilation can be switched off by selecting the "Manual" button on the menu bar above the preview image. The only way of enlarging the preview image is to make the preview pane wider.

The source code can also be viewed as rich text. This gives a partial WYSIWYG feel, which may suit some users. To enable this feature, just click on the "Rich Text" button. The document shown

13.3 Online L^AT_EX Editors

The screenshot shows the Overleaf online LaTeX editor interface. At the top, there's a navigation bar with 'Overleaf' logo, 'Menu', a question mark icon, a gear icon, and 'Nicola Talbot'. Below the menu are tabs: 'Source' (selected), 'Rich Text', and several icons for file operations like copy, paste, find, and more. To the right of these is a status message: 'up-to-date and saved'. The main area is divided into two sections: the left side shows the LaTeX source code, and the right side shows a preview of the document.

Source (Left):

```
1 \documentclass[12pt]{article}
2
3 \usepackage[T1]{fontenc}
4 \usepackage[utf8]{inputenc}
5 \usepackage{lmodern}
6
7
8 \title{Example Document}
9 \author{Nicola Talbot}
10
11 \begin{document}
12 \maketitle
13
14 \begin{abstract}
15 This is an example document. Modified: \pdfcreationdate.
16 \end{abstract}
17
18 \section{Sample Section}
19 \label{sec:sample}
20
21 This is an example section.
22
23 \section{Another Section}
24 \label{sec:another}
25
26 This is another section with a cross-reference to
27 section-\ref{sec:sample}.
28
29 \end{document}
```

Preview (Right):

Example Document
Nicola Talbot
December 19, 2014

Abstract
This is an example document. Modified: D:20141219191042.

1 Sample Section
This is an example section.

2 Another Section
This is another section with a cross-reference to section 1.

Figure 13.3 Overleaf (Mozilla Firefox)

13.3 Online *L^AT_EX* Editors

in Figure 13.3 is reproduced in Figure 13.4 with this feature on. To switch back to viewing the source code, click on the “Source” button. At the time of writing, the rich text function is still in beta.

Another interesting feature is that if I click on an area of the preview pane (for example, on the first section header) the corresponding line in the source code is highlighted. (There’s a short delay between clicking and the source code line being found, so be patient, especially if you have a slow Internet connection.) I wasn’t expecting SyncT_EX support given that the preview window is an image, so I was pleasantly surprised by this.

The free version of Overleaf doesn’t have spell-checking or the ability to change the editor themes, but it does have syntax highlighting.

The list of source files that make up a project can be viewed using the “Menu” button on the main Overleaf navigation bar at the top of the window. Select the “Project” item from this drop-down menu and a window will list the files (shown in Figure 13.5). To add new files, you can click on the “Add files...” button which drops-down a menu where you can choose various options, such as create a new file or folder, or upload from your computer or off-site storage, such as GoogleDocs or Dropbox.

13.3 Online *L^AT_EX* Editors

The screenshot shows the Overleaf online LaTeX editor interface. At the top, there's a navigation bar with 'Overleaf' logo, 'Menu', a question mark icon, a gear icon for settings, and 'Nicola Talbot'. Below the menu are tabs: 'Source' (highlighted), 'Rich Text' (selected), 'Edit', 'Find', 'More', 'Preview' (disabled), 'Manual' (disabled), and 'Auto' (selected). A status message 'up-to-date and saved' is visible.

The main workspace displays a LaTeX document titled 'Example Document' by Nicola Talbot, last modified on December 20, 2014. The document content includes:

- An abstract section with the text "This is an example document. Modified: \pdfcreationdate."
- A bolded "Sample Section" heading followed by the code "\{sec:sample\}".
- The text "This is an example section."
- A bolded "Another Section" heading followed by the code "\{sec:another\}".
- The text "This is another section with a cross-reference to section \ref{\{sec:sample\}}." followed by a reference placeholder.
- The final line of code is "\end{document}".

On the right side, there's a preview pane showing the rendered LaTeX output, which includes the document title, author, date, abstract, and two sections with their respective content.

Figure 13.4 Overleaf—Rich Text Enabled

13.3 Online *L^AT_EX* Editors

You can use the file list to switch the editor to a different source file or you can use the drop-down menu next to the filename to rename or delete the file or perform other operations.

There is also a settings button to the left of the account holder's name on the top navigation bar. The spell check is disabled because it's not available for the free plan, but it's possible to set the editor mode to Vim or Emacs, and the font size can be made smaller or larger.

Share^LA_TE_X I had no problems with the interface on Firefox or Chrome.

As with Overleaf, if you resize the browser, the contents expand or contract with line wrapping in the source rather than introducing a horizontal scrollbar. As you type a command, a drop-down list of suggestions appears. You can click on an item in this list to complete the command name or to fill in the begin and end of an environment. Lines are numbered, but a widget appears next to a line number where an environment begins. You can click on this widget to collapse the environment, hiding its contents from view. This widget is also available for other commands, such as the sectioning commands. When you open a project, it's automatically compiled, but once you edit the source code, you need to manually recompile by clicking on the "Recompile" button.

13.3 Online L^AT_EX Editors

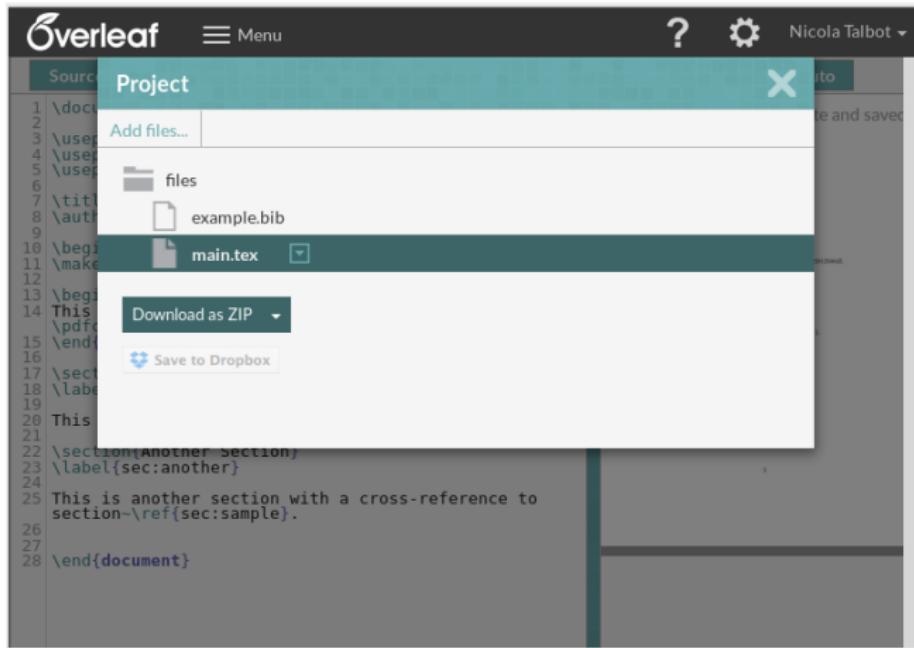


Figure 13.5 Overleaf—Project Files

13.3 Online L^AT_EX Editors

The ShareL^AT_EX project window ([Figure 13.6](#)) has three panels: the file list panel, the source code panel, and the preview panel. The file list panel has buttons that allow you to create new files or folders, upload files, rename files or delete files. You can click on a filename in the list to switch the source code panel to that file. The preview panel has zoom buttons that are displayed when you move the mouse over the top of the preview image. Again SyncT_EX is enabled. You can double-click on an area of the preview image to jump to the corresponding line of code in the middle panel or you can use the right and left arrows between the source code and preview panels.

The style of the source code panel can be changed using the menu widget (the left-most button above the file list panel) which pops up the panel shown in [Figure 13.7](#). (The editor theme is in the lower part of the panel off the edge of the window but can be reached using the vertical scroll bar.) The configurable settings are: compiler (PDFL^AT_EX, L^AT_EX, X_EL^AT_EX or LuaL^AT_EX), main document file, spell check language, auto-complete, theme (various available), key-bindings (none, Vim¹ or Emacs), font size, and PDF viewer (built-in or native). There are also hotkeys that you can use to

¹The Vim key-bindings with the “terminal” editor theme provided an interface similar to my accustomed editing preferences, which I liked.

13.3 Online L^AT_EX Editors

The screenshot shows the ShareLaTeX interface. On the left, the code editor displays a file named "main.tex" containing a LaTeX document. The code includes standard document class and font packages, a title, author information, an abstract section, and two sections with cross-references. On the right, a preview window shows the rendered document with the title "Example Project" and the abstract text. The interface includes a toolbar at the top with various icons for file operations and a "Recompile" button.

```
1 \documentclass{article}
2
3 \usepackage[utf8]{inputenc}
4 \usepackage[T1]{fontenc}
5 \usepackage{lmodern}
6
7 \title{Example Project}
8 \author{(Nicola Talbot)}
9
10 \begin{document}
11 \maketitle
12
13 \begin{abstract}
14 This is an example document. Modified:
15 \pdfcreationdate.
16 \end{abstract}
17
18 \section{Sample Section}
19 \label{sec:sample}
20
21 This is an example section.
22
23 \section{Another Section}
24 \label{sec:another}
25
26 This is another section with a
27 cross-reference to
28 section \ref{sec:sample}.
29
30 \end{document}
31
```

Example Project
Nicola Talbot
December 20, 2014

Abstract
This is an example document. Modified: \pdfcreationdate.
1 Sample Section
This is an example section.
2 Another Section
This is another section with a cross-reference to section 1.

Figure 13.6 ShareL^AT_EX (Mozilla Firefox)

13.3 Online *L^AT_EX* Editors

navigate your way around the source code. These can be displayed by clicking on the “Show Hotkeys” button below the settings list. To close the settings panel, just click on one of the other panels.

Error Handling

To test the error handling, I misspelt a command. (I removed the “f” from the command `\pdfcreationdate`.)

Overleaf The line where the error occurs is highlighted and a popup window displays the message immediately below the line (see [Figure 13.8](#)). Moving the cursor away from the error line will make the message disappear. Moving the cursor back to the error line will make the message reappear. At the right-hand end of the menu bar above the preview pane, there’s a red drop-down menu labelled “compile error”. There are two items: “go to first error/warning” and “open latex log”. I couldn’t find any way to open the log file when there weren’t any errors or warnings. (There are some useful bits of information in log files that aren’t error or warning messages, which I sometimes like to check even if the document appears okay.)

13.3 Online L^AT_EX Editors

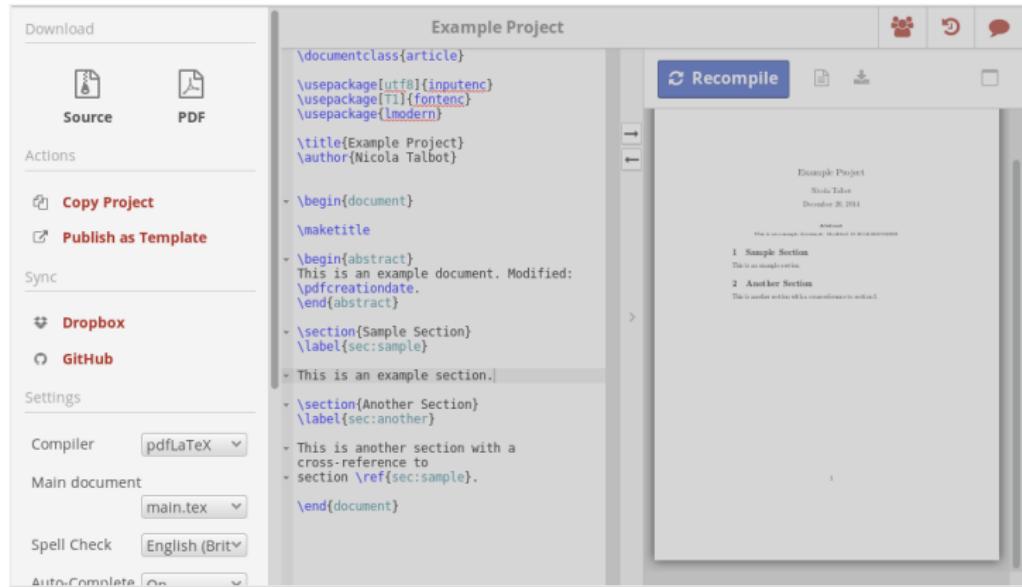


Figure 13.7 ShareL^AT_EX—Settings

13.3 Online L^AT_EX Editors

The screenshot shows the Overleaf interface with a LaTeX document open. The code editor on the left contains the following LaTeX code:

```
1 \documentclass[12pt]{article}
2
3 \usepackage[T1]{fontenc}
4 \usepackage[utf8]{inputenc}
5 \usepackage{lmodern}
6
7
8 \title{Example Document}
9 \author{Nicola Talbot}
10
11 \begin{document}
12 \maketitle
13
14 \begin{abstract}
15 This is an example document. Modified: \pdcreationdate.
16
17 Undefined control sequence.
18 l.15 ...xample document. Modified: \pdcreationdate
19
20 .
21
22
23 \section{Another Section}
24 \label{sec:another}
25
26 This is another section with a cross-reference to
27 section-\ref{sec:sample}.
28
29 \end{document}
```

A red box highlights the line `This is an example document. Modified: \pdcreationdate.`. A tooltip above the red box says "compile error" and lists two options: "go to first error / warning" and "open latex log". The right side of the screen shows the rendered document with sections 1 and 2, and a note about the abstract section.

Figure 13.8 Overleaf—Error Handling

13.3 Online L^AT_EX Editors

ShareL^AT_EX The line where the error occurs has a small red box with a white cross in it next to the line number. If you move the mouse over this box, the error message will be displayed. You can also click on the “Log and output files” button above the preview window to display more details about the error (as shown in Figure 13.9). Below the detailed error message are buttons that allow you to view the full transcript, clear cached files and a drop-down menu that allows you to view the auxiliary files.

T_EX Engine

Overleaf I was interested to discover that Overleaf seems to be able to determine what engine to use when compiling the document. The example document shown in Figure 13.3 uses `\pdfcreationdate`, which is a PDFT_EX primitive. Inspecting the log file shows that PDFL^AT_EX is being used and the command successfully produces the date stamp. However, if I add the `pstricks` package to this document, I get an “Undefined control sequence” error for this command. Investigating the log file shows that X_EL^AT_EX is being used (which doesn’t provide `\pdfcreationdate`).

ShareL^AT_EX With ShareL^AT_EX you need to explicitly set the engine you want

13.3 Online L^AT_EX Editors

The screenshot shows the ShareLaTeX online editor interface. On the left, the code editor displays a LaTeX document named "main.tex". The code includes standard document class and font packages, a title, author, and abstract section. Line 16 contains the command \pdcreationdate, which is highlighted in red and underlined, indicating it is undefined. A tooltip or error message box appears over this line, stating: "Undefined control sequence. main.tex, line 16. I16 ...xample document. Modified: \pdcreationdate". Below the code editor, there are buttons for "Recompile" and "View Raw Logs". The right side of the interface has a sidebar with icons for sharing, reporting bugs, and other project management features.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[modern]{font}
\title{Example Project}
\author{(Nicola Talbot)}
\begin{document}
\maketitle
\begin{abstract}
This is an example document. Modified: \pdcreationdate.
\end{abstract}
\section{Sample Section}
\label{sec:sample}
\section{Another Section}
\label{sec:another}
\end{document}
```

Figure 13.9 ShareL^AT_EX — Error Handling

13.3 Online L^AT_EX Editors

to use. You have a choice of PDFL^AT_EX, L^AT_EX, X_{EL}L^AT_EX and LuaL^AT_EX. The default is PDFL^AT_EX.

Available Classes and Packages

I tried out the classes and packages described in this book. Unsurprisingly, packages that aren't on CTAN, such as pgfornament, also aren't available with the online editors nor are some packages that require external scripts, such as piechart and vc. Neither site had the non-T_EX Live packages eforms, draftmark, pgf-pie and rtsched. Inspecting the generated log files showed that both sites use T_EX Live on a Unix-like system.

In addition to the above, Overleaf didn't have drm or gitinfo2 installed (but it did have gitinfo2's predecessor, gitinfo) nor did it have the pressrelease class. (Both drm and pressrelease were released in September 2014, three months before the time of writing this, and gitinfo2 was new in May 2014.) ShareL^AT_EX had drm, gitinfo2 and pressrelease installed.

Therefore it seems that, at the time of writing, both Overleaf and ShareL^AT_EX have reasonably up-to-date T_EX Live distributions, but ShareL^AT_EX's distribution is more up-to-date. This may, of course, have changed since I tested both sites.

External Applications

To test commonly-used external applications, I added an example bib file (called `example.bib`) that contained a sample entry:

```
@book{sample,  
    title = "A Sample Book",  
    author = "Ann Author",  
    publisher = "A Publisher",  
    year = 2014  
}
```

I added this sample citation to my document and also added an index (with the `makeidx` package), and a glossary and list of acronyms (with the `glossaries` package).

Both Overleaf and ShareL^AT_EX correctly generated the bibliography, index, glossary and list of acronyms. I didn't need to specify any of the external applications, both sites automatically ran `bibtex` and `makeindex` on the appropriate files.

Interaction Between Collaborators

Overleaf I viewed my example source file in both Firefox and Chrome at the same time. As I edited the text in one window, the other window was automatically updated, so it seems that all collaborators should always be viewing the latest version of the code.

Each collaborator can add comments to the code using the “Add a comment” button (it looks like a speech bubble with a plus inside it). I added a comment, and the source code pane was switched to rich text format, as shown in Figure 13.10. The comment can be hidden using the “Hide Comment” button on the top left of the comment box. When the comment is no longer needed, it can be closed using the “close” link at the bottom of the comment box or another author can reply to the comment using the “reply” link. If you switch back from the rich text mode to the regular source code mode, the comment is present using T_EX’s standard commenting mechanism with the % character, but it also includes the commenter’s email address and the date the comment was made.

ShareL^AT_EX The free plan that I tried out didn’t support multiple collaborators.

13.3 Online *L^AT_EX* Editors

The screenshot shows the Overleaf online LaTeX editor interface. At the top, there's a navigation bar with 'Overleaf' logo, 'Menu', a question mark icon, settings, and user 'Nicola Talbot'. Below the menu are tabs: 'Source' (selected), 'Rich Text' (highlighted in green), 'Edit', 'Find', 'More', 'Preview' (with 'Manual' and 'Auto' sub-options), and a status message 'up-to-date and saved'. The main workspace contains a LaTeX document titled 'Example Document' by Nicola Talbot, dated December 20, 2014. The document includes an abstract, sample sections, and references. A prominent feature is a 'Rich Text' comment box from Nicola Talbot, timestamped '5 minutes ago', which reads: 'This is a \gls{sample} document. Modified: \pdfcreationdate.' Below the comment box are 'close' and 'reply' buttons. The right side of the screen shows a preview pane with the document's structure and a small 'Index' section at the bottom.

Figure 13.10 Overleaf—Comment Viewed as Rich Text

Accessibility

Given the complexity and visual nature of both sites, I didn't expect either site to work well with a simple text-only browser such as Lynx, but that and the Konqueror Jovie plugin were the only means I had to test accessibility for the visually impaired.

Overleaf The site doesn't appear to provide alternative text for images. For example, on the [pricing and plans page](#) the ticks and crosses that indicate which options are available with which plans are invisible to the text-to-speech synthesizer, which is useless for any users who can't see the images. On the other hand, text that isn't visually displayed on the page is read out, which is a bit confusing.

The project window doesn't render correctly in either Lynx or Konqueror and I wasn't able to view or edit the document source code. It's possible that a more sophisticated screen reader, such as Jaws, can produce better results with a different browser.

ShareL^AT_EX Again there didn't appear to be any alternative text for images. When viewed in Lynx, the prices weren't visible on the [Plans and Pricing](#) page, however they were read out by Konqueror's Jovie plugin. I couldn't view the project window in Konqueror. I just

13.3 Online *LATEX* Editors

got an error claiming that the project had been modified by collaborators and that I should ask the project owner to upgrade the account. Again, it's possible that a more sophisticated speech reader can produce better results with a different browser.

Mobile Devices

I tried both sites on my Arnova 7GB tablet. This is a very basic mobile device and my rural broadband is a little iffy, so I didn't expect a fast response from either site. I also borrowed my brother's iPad and my son's "it tablet" and tried both sites on those.

Both sites worked fine with the iPad and the it tablet, but they didn't work well with the Arnova tablet.

Overleaf I was able to login with the Arnova's default web browser and with Opera Mobile, but whenever I tried to access my list of projects from the dashboard, I was redirected to the site's home page.

ShareLATEX The Arnova's default web browser and also the Opera Mobile browser were unable to render the email and password fields in ShareLATEX's login page, so I was unable to test out the site as I was unable to login.

Summary

Both sites are impressive and have similar interfaces, but they are not browser-independent, only fully-functioning with the big well-known browsers (which is not really surprising given the complexity of the sites). If you have collaborators who have difficulty using complicated web interfaces (for example, if they are visually-impaired) you may want to consider using a version control repository instead so that they can use their preferred text editor and other tools that are accessible for them.

Share \LaTeX 's paid plans have the ability to sync to GitHub, but remember that private GitHub repositories also require payment. Both sites have paid plans that also provide the ability to save to Dropbox. Overleaf's paid plans are cheaper than Share \LaTeX 's.

If any of your collaborators are wary of \LaTeX and are easily confused by the document commands, they may be swayed by Overleaf's rich text view if all they need to do is write text (as opposed to complicated equations or diagrams).

Share \LaTeX 's free plan comes with a spell checker, multiple editor themes and auto-complete, but Overleaf's free plan doesn't. (The paid plans do.) On the other hand, Share \LaTeX 's free plan doesn't allow multiple collaborators whereas Overleaf's free plan does.

Both sites displayed the time stamp in UTC+0, which also happens to

13.3 Online L^AT_EX Editors

be my current time zone (GMT) at the time of writing. This surprised me as I was expecting at least one of the sites to use a server in a different time zone. I can't tell if they are actually in the same time zone as me or if they can determine my location and set the time zone at the start of the document compilation.

ACKNOWLEDGEMENTS

Thank you to Joseph Wright for proof-reading this book and providing useful feedback. Thank you to my son, Cameron, for providing the Sumatra and Foxit on Windows screen shots, and for lending his tablet to try out the online editors. Thank you to my brother Stephen Talbot for lending his iPad to try out the online editors. Thank you to Paulo Cereda for being so patient with my bug reports and awkward feature requests for [arara!](#)! Thank you to the package and class authors who responded to my bug reports. Thank you to Barbara Beeton for spotting errors and giving useful advice, and to everyone else who provided feedback during the creation of this book, and thank you to all those who keep the \TeX community going, including the [CTAN](#) team, [TUG](#) and local \TeX user groups, and all the class and package authors who share the fruits of their labours! And, of course, thank you to Donald Knuth for creating \TeX , Leslie Lamport for creating \LaTeX , the \LaTeX team for taking over the development of \LaTeX , and the developers of $\text{PDF}\text{\TeX}$, $\text{Xe}\text{\TeX}$ and $\text{Lua}\text{\TeX}$.

If you're not already a member of [TUG](#) or a local \TeX user group, please consider joining.

BIBLIOGRAPHY

- [1] American Mathematical Society (AMS). User's guide for the `amsmath` package (version 2.0), 2002. [CTAN Mirror](#) or [texdoc amsmath](#).
- [2] Donald Arseneau. The `ulem` package, 2011. [CTAN Mirror](#) or [texdoc ulem](#).
- [3] Donald Arseneau. The `framed` package, 2012. [CTAN Mirror](#) or [texdoc framed](#).
- [4] Donald Arseneau. `shapepar.sty` v2.2, 2013. [CTAN Mirror](#) or [texdoc shapepar](#).
- [5] Donald Arseneau. `url.sty` version 3.4, 2013. [CTAN Mirror](#) or [texdoc url](#).
- [6] Brian D. Beitzel. Formatting meeting minutes with the `meetingmins` L^AT_EX class, 2013. [CTAN Mirror](#) or [texdoc meetingmins](#).

Bibliography

- [7] Johannes L. Braams and Javier Bezos. babel version 3.9k, 2014. [CTAN Mirror](#) or [texdoc babel](#).
- [8] Klaus Dieter Braune and Richard Gussmann. Standard document class ‘dinbrief’, 2007. [CTAN Mirror](#) or [texdoc dinbrief](#).
- [9] Achim D. Brucker. The svninfo package version 0.7.4, 2010. [CTAN Mirror](#) or [texdoc svninfo](#).
- [10] David Carlisle. The ifthen package, 2001. [texdoc ifthen](#).
- [11] David Carlisle. The longtable package, 2004. [CTAN Mirror](#) or [texdoc longtable](#).
- [12] David Carlisle. The keyval package, 2014. [texdoc keyval](#).
- [13] David Carlisle. The tabularx package, 2014. [texdoc tabularx](#).
- [14] David Carlisle and The L^AT_EX3 Project. Packages in the ‘graphics’ bundle, 2014. [CTAN Mirror](#) or [texdoc graphics](#).
- [15] Florent Chervet. The etextools macros, 2010. [CTAN Mirror](#) or [texdoc etextools](#).

Bibliography

- [16] The `comp.text.tex` news group. <http://groups.google.com/group/comp.text.tex>.
- [17] Oliver Corff. invoice 0.9: a package for writing invoices, 2011. [CTAN Mirror](#) or [texdoc invoice](#).
- [18] The comprehensive TeX archive network. <http://ctan.org/>.
- [19] Patrick W. Daly. Natural sciences citations and references, 2010. [CTAN Mirror](#) or [texdoc natbib](#).
- [20] Marco Daniel and Elke Schubert. The `mdframed` package: auto-split frame environment, 2013. [CTAN Mirror](#) or [texdoc mdframed](#).
- [21] Wybo Dekker. The `isodoc` class for letters, invoices, and more, 2014. [CTAN Mirror](#) or [texdoc isodoc](#).
- [22] Jean-Pierre F. Drucbert. The `minitoc` package, 2008. [CTAN Mirror](#) or [texdoc minitoc](#).
- [23] Thomas Emmel. `ticket.sty`: making labels, visiting cards, pins and flash-cards with L^AT_EX, 2010. [CTAN Mirror](#) or [texdoc ticket](#).

Bibliography

- [24] Simon Fear. Publication quality tables with L^AT_EX, 2005. [CTAN Mirror](#) or [texdoc booktabs](#).
- [25] Jürgen Fenn. The T_EX catalogue online, topic index. [CTAN Mirror](#).
- [26] Christian Feuersänger. Manual for package pgfplots: 2D/3D plots in L^AT_EX, version 1.11, 2014. [CTAN Mirror](#) or [texdoc pgfplots](#).
- [27] Denis Girou and Herbert Voß. pst-gantt v.0.22, 2013. [CTAN Mirror](#) or [texdoc pst-gantt](#).
- [28] Jeff Goldberg. MS-Word is Not a document exchange format. <http://www.goldmark.org/netrants/no-word/attach.html>, 2005. Accessed: 2014-11-10.
- [29] Donald P. Goodman III. The drm font package, 2015. [CTAN Mirror](#) or [texdoc drm](#).
- [30] GNU on Windows. <http://wiki.github.com/bmatzelle/gow/>. Accessed: 2015-09-05.
- [31] Thorsten Hansen. The bibunits package, 2004. [CTAN Mirror](#) or [texdoc bibunits](#).

Bibliography

- [32] Thorsten Hansen. The multibib package, 2008. [CTAN Mirror](#) or [texdoc multibib](#).
- [33] Patrick Happel. lipsum—access to 150 paragraphs of *Lo&rem ip&sum* dummy text, 2014. [CTAN Mirror](#) or [texdoc lipsum](#).
- [34] Carsten Heinz, Brooks Moses, and Jobst Hoffmann. The listings package, 2014. [CTAN Mirror](#) or [texdoc listings](#).
- [35] Stephan Hennig. The vc bundle, 2008. [CTAN Mirror](#) or [texdoc vc-manual](#).
- [36] Philip S. Hirschhorn. Using the exam document class, 2011. [CTAN Mirror](#) or [texdoc exam](#).
- [37] Neil Chue Hong. Choosing a repository for your software project. <http://software.ac.uk/resources/guides/choosing-repository-your-software-project>, 2013. Accessed: 2014-12-12.
- [38] Information Commissioner's Office. Data protection and freedom of information advice. <http://ico.org.uk/>. Accessed: 2014-09-24.
- [39] Bogusław Jackowski and Janusz Marian Nowacki. The Latin Modern family of fonts, 2009. [CTAN Mirror](#) or [texdoc lmodern](#).

Bibliography

- [40] Alan Jeffrey and Frank Mittelbach. `inputenc.sty` v1.2b, 2014. [CTAN Mirror](#) or [texdoc inputenc](#).
- [41] Uwe Kern. Extending L^AT_EX's color facilities: the `xcolor` package, 2007. [CTAN Mirror](#) or [texdoc xcolor](#).
- [42] Axel Kielhorn. The `wasysym` macro package for L^AT_EX 2_E v2.0, 2003. [CTAN Mirror](#) or [texdoc wasysym](#).
- [43] Sebastian Marius Kirsch. `bizcard`: a L^AT_EX 2_E package for business/visiting/calling cards, 1999. [CTAN Mirror](#) or [texdoc bizcard](#).
- [44] Ekkart Kleinod. The `changes`-package: Manual change markup—version 2.0.3, 2014. [CTAN Mirror](#) or [texdoc changes.english](#).
- [45] Ingo Klöckl. Paket `ifsym` und die `if...-fonts` (German), 2001. [CTAN Mirror](#) or [texdoc ifsym](#).
- [46] Donald Ervin Knuth. *The T_EXbook*. Addison-Wesley, 1986.
- [47] Markus Kohm and Jens-Uwe Morawski. KOMA-Script a versatile L^AT_EX 2_E bundle, 2014. [CTAN Mirror](#) or [texdoc koma](#).

Bibliography

- [48] Tobias Kuhn. *bchart*: Simple bar charts in L^AT_EX, 2012. [CTAN Mirror](#) or [texdoc bchart](#).
- [49] The L^AT_EX font catalogue. <http://www.tug.dk/FontCatalogue/>.
- [50] The L^AT_EX3 Project. The L^AT_EX3 interfaces, 2014. [texdoc interface3](#).
- [51] Philipp Lehman and Joseph Wright. The *etoolbox* package: an e-TeX toolbox for class and package authors, 2011. [CTAN Mirror](#) or [texdoc etoolbox](#).
- [52] Richard Lewis. The *svn* package, 2007. [CTAN Mirror](#) or [texdoc svn](#).
- [53] Knut Lickert. L^AT_EX2_E for people in associations *minutes.sty*, 2009. [CTAN Mirror](#) or [texdoc minutes](#).
- [54] Giuseppe Lipari. The *rtsched* package for L^AT_EX (version 1.0), 2011. [CTAN Mirror](#) or [texdoc rtsche](#).
- [55] Brent Longborough. *gitinfo2.sty*: a package for accessing metadata from the git DVCS version 2.0.4, 2014. [CTAN Mirror](#) or [texdoc gitinfo2](#).

Bibliography

- [56] Gonzalo Medina. The `background` package, 2014. [CTAN Mirror](#) or [texdoc background](#).
- [57] Michael Mehlich. `fp` package, 1999. [CTAN Mirror](#) or [texdoc fp](#).
- [58] Frank Mittelbach and David Carlisle. A new implementation of L^AT_EX's tabular and array environment, 2014. [CTAN Mirror](#) or [texdoc array](#).
- [59] Frank Mittelbach, Robin Fairbairns, Werner Lemberg, and L^AT_EX3 Project Team. L^AT_EX font encodings, 2014. [texdoc encguide](#).
- [60] Ahmed Musa. The `xwatermark` package, 2012. [CTAN Mirror](#) or [texdoc xwatermark](#).
- [61] MySQL::market share. <http://www.mysql.com/why-mysql/marketshare/>.
- [62] Clemens Niederberger. The ExSheets bundle, 2014. [CTAN Mirror](#) or [texdoc exsheets](#).
- [63] Rolf Niepraschk and Hubert Gässlein. The `pst-pdf` package, 2008. [CTAN Mirror](#) or [texdoc pst-pdf](#).

Bibliography

- [64] Rolf Niepraschk, Walter Schmidt, and Hubert Gäßlein. The document class leaflet, 2013. [CTAN Mirror](#) or [texdoc](#) leaflet-manual.
- [65] Scott Pakin. The comprehensive L^AT_EX symbol list, 2009. [CTAN Mirror](#) or [texdoc](#) symbols.
- [66] C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version Control with Subversion*. O'Reilly, 2008. <http://svnbook.red-bean.com/>.
- [67] CV Radhakrishnan, CV Rajagopal, and Antoine Chambert-Loir. Trivial experiments with psTricks manipulation, 2003. [CTAN Mirror](#) or [texdoc](#) pdftricks.
- [68] Sebastian Rahtz, Leonor Barroca, Grant Gustafson, and Julian Gilbey. A package for making sticky labels in L^AT_EX, 2003. [texdoc](#) labels.
- [69] Sebastian Rahtz and Heiko Oberdiek. Hypertext marks in L^AT_EX: a manual for hyperref, 2012. [CTAN Mirror](#) or [texdoc](#) hyperref.
- [70] Luis Rández and Juan I. Montijano. The ticking digital clock tdclock package v2.3, 2014. [CTAN Mirror](#) or [texdoc](#) tdclock.

Bibliography

- [71] George Reese, Randy Jay Yarger, and Tim King. *Managing & Using MySQL*. O'Reilly, 2nd edition, 2002.
- [72] Axel Reichert. `currvita.sty`, 1999. [CTAN Mirror](#) or [texdoc currvita](#).
- [73] Alan Ristow. `pst-bar` v.0.92: bar charts for pstricks, 2008. [CTAN Mirror](#) or [texdoc pst-bar](#).
- [74] R. M. Ritter. *Oxford Style Manual*. Oxford University Press, 2003.
- [75] Will Robertson. A couple of things involving environments (`environ` v0.3), 2014. [CTAN Mirror](#) or [texdoc environ](#).
- [76] Will Robertson and Khaled Hosny. The `fontspec` package: Font selection for X^{\LaTeX} and Lua^{\LaTeX}, 2014. [CTAN Mirror](#) or [texdoc fontspec](#).
- [77] Jonathan Sauer. The `collect` package, 2004. [CTAN Mirror](#) or [texdoc collect](#).
- [78] Bernd Schandl. `paralist` extended list environments, 2013. [CTAN Mirror](#) or [texdoc paralist](#).

Bibliography

- [79] Martin Scharrer. The `svn-prov` package: Use SVN Id keywords for package, class and file header (version 3.1862d), 2010. [CTAN Mirror](#) or [texdoc svn-prov](#).
- [80] Martin Scharrer. The `rcs-multi` package v0.1a, 2011. [CTAN Mirror](#) or [texdoc rcs-multi](#).
- [81] Martin Scharrer. The `svn-multi` package version 2.4d, 2011. [CTAN Mirror](#) or [texdoc svn-multi](#).
- [82] Martin Scharrer. The `standalone` package, 2012. [CTAN Mirror](#) or [texdoc standalone](#).
- [83] Martin Scharrer. The `mwe` package, 2012. [CTAN Mirror](#) or [texdoc mwe](#).
- [84] Walter Schmidt. Using common PostScript fonts with L^AT_EX, 2004. [CTAN Mirror](#) or [texdoc psnfss](#).
- [85] Rainer Schöpf, Bernd Raichle, and Chris Rowley. A new implementation of L^AT_EX's `verbatim` and `verbatim*` environments, 2001. [CTAN Mirror](#) or [texdoc verbatim](#).

Bibliography

- [86] Joachim Schrod. The rcs package, 1995. [CTAN Mirror](#) or [texdoc rcs](#).
- [87] Steven B. Segletes. The censor package, 2013. [CTAN Mirror](#) or [texdoc censor](#).
- [88] Wolfgang Skala. The pgfgantt package v4.0, 2013. [CTAN Mirror](#) or [texdoc pgfgantt](#).
- [89] Richard M. Smith. Microsoft Word bytes Tony Blair in the butt. <http://www.computerbytesman.com/privacy/blair.htm>, 2003. Accessed: 2014-11-10.
- [90] R. Stepanyan. The bardiaq package, 2003. [CTAN Mirror](#) or [texdoc bardiaq](#).
- [91] D. P. Story. AcroTEX.Net eforms and insdls documentation v2.0, 2013. [CTAN Mirror](#) or [texdoc eforms](#).
- [92] Nicola L. C. Talbot. Writing a L^AT_EX class file to produce a form. *The LaTeX Community's Know How Section*, November 2009. <http://www.latex-community.org/know-how/latex/54-latex-document-classes/342-writing-a-latex-class-file-to-produce-a-form>.

Bibliography

- [93] Nicola L. C. Talbot. *L^AT_EX for Complete Novices*, volume 1 of *Dickimaw L^AT_EX Series*. Dickmaw Books, 2012. <http://www.dickimaw-books.com/latex/novices/>.
- [94] Nicola L. C. Talbot. `probsoln`: creating problem sheets optionally with solutions, 2012. [CTAN Mirror](#) or [texdoc probsoln](#).
- [95] Nicola L. C. Talbot. User manual for `datatool` bundle, 2013. [CTAN Mirror](#) or [texdoc datatool-user](#).
- [96] Nicola L. C. Talbot. *Using L^AT_EX to Write a PhD Thesis*, volume 2 of *Dickimaw L^AT_EX Series*. Dickmaw Books, 2013. <http://www.dickimaw-books.com/latex/thesis/>.
- [97] Nicola L. C. Talbot. `pressrelease` v1.0: typesetting press releases, 2014. [CTAN Mirror](#) or [texdoc pressrelease](#).
- [98] Nicola L. C. Talbot. Creating flow frames for posters, brochures or magazines using `flowfram.sty` version 1.16, 2014. [CTAN Mirror](#) or [texdoc ffuserguide](#).
- [99] Nicola L. C. Talbot. `mfirststuc.sty` v2.0: uppercasing first letter, 2015. [CTAN Mirror](#) or [texdoc mffirststuc](#).

Bibliography

- [100] Nicola L. C. Talbot. `datetime2-english` v1.01: English module for `date-time2` package, 2015. [CTAN Mirror](#) or [texdoc](#) `datetime2-english`.
- [101] Nicola L. C. Talbot. `datetime2` v1.0: date and time formats, 2015. [CTAN Mirror](#) or [texdoc](#) `datetime2`.
- [102] Till Tantau. The `TikZ` and `PGF` packages: Manual for version 3.0.0, 2013. [CTAN Mirror](#) or [texdoc](#) `pgf`.
- [103] Till Tantau, Joseph Wright, and Vedran Miletic. The `beamer` class: user guide for version 3.33, 2013. [CTAN Mirror](#) or [texdoc](#) `beamer`.
- [104] Jamal K. Tartir. `LATEX` package `jlabels`, 2011. [CTAN Mirror](#) or [texdoc](#) `jlabels`.
- [105] Bob Tennent. Alegreya fonts with `LATEX` support, 2015. [CTAN Mirror](#) or [texdoc](#) `alegreya`.
- [106] Hn Thn Thành, Sebastian Rahtz, Hans Hagen, Hartmut Henkel, Paweł Jackowski, and Martin Schröder. The `pdfTEX` user manual, 2014. [CTAN Mirror](#) or [texdoc](#) `pdftex`.
- [107] Paul A. Thompson. `newlfm.cls`: a new letter, fax, memo document class for `LATEX 2 ϵ` , 2009. [CTAN Mirror](#) or [texdoc](#) `newlfm`.

Bibliography

- [108] The TeX user group. <http://tug.org/>.
- [109] UK list of TeX frequently asked questions (UK TeX FAQ). <http://www.tex.ac.uk/faq> or `texdoc faq`.
- [110] Hideo Umeki. The geometry package v5.6, 2010. [CTAN Mirror](#) or `texdoc geometry`.
- [111] Boris Veytsman. Printing envelopes and labels in L^AT_EX 2_E: envlab package user guide, 1997. [CTAN Mirror](#) or `texdoc elguide`.
- [112] Nicola Vitacolonna. europecv: an unofficial class for european curricula, 2006. [CTAN Mirror](#) or `texdoc europecv`.
- [113] Jürgen Vollmer. The rcsinfo package, 2005. [CTAN Mirror](#) or `texdoc rcsinfo`.
- [114] Herbert Voß. pst-barcode: a PSTricks package for drawing barcodes, 2013. [CTAN Mirror](#) or `texdoc pst-barcode`.
- [115] Hans-Christoph Wirth. formular.sty v1.0a, 2015. [CTAN Mirror](#) or `texdoc formular`.

Bibliography

- [116] Yuan Xu. Drawing pie chart by using pgf-pie v0.2, 2012. [CTAN Mirror](#) or [texdoc pgf-pie](#).
- [117] Timothy Van Zandt. PSTricks PostScript macros for generic TeX, 2007. [CTAN Mirror](#) or [texdoc pst-user](#).

GLOSSARY

Active Character A character with [category code 13](#). An active character is interpreted by \TeX as a macro but it isn't escaped with a leading backslash. By default, there's only one active character \sim but some packages, such as `babel`, make other characters active. In particular the `inputenc` package makes all non-Latin or accented characters (such as é) active.

arara A [Java](#) application that automates the process of building a \LaTeX document. See also [Volume 2](#) [96, §1.1.2]. URL: <http://ctan.org/pkg/arara>

Basic Latin Character One of the letters `a`, ..., `z`, `A`, ..., `Z`. (Part of the 7-bit ASCII set.) See also [extended Latin character](#).

Boolean Key A key in a [key=value list](#) where the value may be either `true` or `false`. The value part may be omitted if it's `true`. For example, `noheader=true` and `noheader` both switch on the `noheader` setting.

Glossary

Category Code The code assigned to a character that identifies its category.

For example, the character “a” has the category code 11, which means it’s a letter and can be used to form control words (see [Volume 1 \[93, §2.6\]](#)). The character “\$” has the category code 3, which means it’s the math-shift character. It’s possible to change the category code of a character. For example, in `.cls` (class) or `.sty` (package) files, the "@" character has the category code 11 (“letter”) so it can be used in control words (such as `\@for`) but outside of those files the "@" character has the category code 12 (“other”), so it can be used in the [control symbol](#) `\@` but not in any [control words](#). For more details, see [The TeXbook \[46\]](#).

Comma-separated List A list where each item is separated by a comma.

Leading or trailing spaces on either side of individual items in the list may or may not be ignored, depending on the context. If in doubt, err on the side of caution and remove spaces or suppress [EOL](#) terminators with the `%` comment character.

CSV Comma-separated variable.

CTAN The Comprehensive TeX Archive Network [\[18\]](#). <http://mirror.ctan.org/>.

Glossary

CV Curriculum vitae (plural: curricula vitae).

datatooltk A Java application that can be used to edit datatool databases. It can also be used to import data from CSV files, Excel spreadsheets, SQL databases and TeX files that can be loaded using probsoln's \loadallproblems command. This application can either be run in batch or GUI mode. Remember to add the datatooltk/bin directory to your system path if you want to invoke it from a command prompt. The way to do this varies according to your operating system. If you don't know how to do it, try doing a web search for "set path environment". URL: <http://www.dickimaw-books.com/software/datatooltk/>

datatooltk-gui A script that invokes **datatooltk** in GUI mode.

EOL End of line.

A character, or sequence of characters, signifying a line break (in the source code not in the resulting PDF) created by pressing the enter or return  key. The underlying character code is dependent on the operating system. For example, on Unix the EOL is indicated by the line feed (\n or 0x0A) symbol (LF) whereas on Microsoft Windows the EOL is indicated by a combination of the carriage return (\r or 0x0D) symbol (CR) and the LF symbol. The

Glossary

category code for an EOL is 5, but usually TeX treats an EOL the same as the space symbol, unless it's immediately followed by another EOL, in which case a paragraph break created.

Extended Character A character that's outside of the 7-bit ASCII set. For example, an **extended Latin character** or any character from a non-Latin alphabet. Note that the numerical code representing a non-Latin character varies according to the file's input encoding. See also **extended Latin character**.

Extended Latin Character A character that's created by combining **basic Latin characters** to form ligatures (e.g. æ) or by applying diacritical marks to a **basic Latin character** or characters (e.g. á or ø). (See also **extended character**.) Note that the numerical code representing an extended Latin character varies according to the file's input encoding. For example, ï has codepoint 0x00EF in the Unicode table, but has code 139 (0x8B) in the ASCII table.

flowframtk A Java application that can be used to construct frames for the flowfram package. URL: <http://www.dickimaw-books.com/software/flowframtk/>

GOW GNU On Windows [30]. <https://github.com/bmatzelle/gow>.

Glossary

GUI Graphical user interface.

ICO Information Commissioner's Office [38]. <http://ico.org.uk/>.

Internal Command A command that contains an at character (@) in its name (such as `\@for`). These commands are intended for internal use in class or package files. You should avoid using them within your document, but if you really need to, you must first change the category code of the @ symbol to "letter" (via `\makeatletter`) and, after you've used the internal command, change the category code back to "other" (via `\makeatother`).

Java A language that can be deployed in a cross-platform environment, which means that as long as you have the Java runtime environment installed you can run a Java application regardless of the operating system used to develop the application. URL: <https://java.com/>

Key=value List A comma-separated list of $\langle key \rangle = \langle value \rangle$ entries. Spaces on either side of the $\langle key \rangle$ and $\langle value \rangle$ are usually ignored. If $\langle value \rangle$ contains a comma (for example, the value is a list), the value must be enclosed in braces and unwanted spaces should be removed. (Recall from Volume 1 [93, §2] that unwanted space caused by the `EOL` character can be ignored using the `%` comment character.)

Glossary

kpsewhich An application used for Kpathsea lookup and expansion.. URL:
<http://tug.org/texinfohtml/kpathsea.html>

[FAQ: Which tree to use]

make An application for automating the building of software or documents by specifying dependencies

Primitive A core command that isn't defined in terms of other commands.

The *T_EXbook* [46] describes primitives as low-level atomic operations that can't be decomposed into simpler functions. There are around 300 primitives provided by T_EX. Other T_EX formats, such as PDFT_EX or X_ET_EX provide additional primitives.

Shell Escape The ability to spawn processes during the document build. Since this is a security risk, it's usually disabled by default or a restricted version may be enabled that only allows a pre-set list of commands to be run.

[FAQ: Spawning programs from (La)TeX: \write18]

Special Character A character that has a special meaning to T_EX. The common special characters are: `\` (the escape character, category code 0) `{` (group beginning, category code 1) `}` (group ending, category code 2) `$` (math-shift, category code 3) `&` (alignment tab, category code 4) `#` (parameter, category code 6) `^` (superscript, category code 7) `_` (subscript, category code 8) `%` (comment, category code 9)

Glossary

and `~` (an active character). In some cases, the whitespace characters `EOL` (category code 5) and `_` (category code 10) may also be considered special.

SQL Structured query language.

texdef A Perl script that displays the definition of (La)TeX commands.
URL: <http://ctan.org/pkg/texdef>

Token Either a character (including **special characters**) or a control sequence.

TUG TeX User Group [108]. <http://tug.org/>.

UK FAQ UK List of TeX Frequently Asked Questions [109]. <http://www.tex.ac.uk/faq>.

UTF-8 Unicode Transformation Format—8-bit.

This is an encoding that can represent every character in the Unicode character set (see also **extended character** and **extended Latin character**). If your source code contains Unicode characters, such as é (codepoint `0x00E9`) or ø (codepoint `0x00F8`), you must set both

Glossary

your editor to UTF-8 and use the `inputenc` package with the `utf8` option in your document: `\usepackage[utf8]{inputenc}`. (Note that it's recommended that you also load `fontenc` if you use `inputenc`. See [Volume 1 \[93, §4.3.1\]](#).) X^ET_EX users should just load `fontspec` instead of `inputenc` and `fontenc`.

If you are using TeXworks, the status bar at the bottom of the window should show the encoding. You can change the editor encoding in TeXworks via the `Edit→Preferences` menu and select the “Editor” tab.

Whitespace A whitespace character is an invisible character that represents horizontal or vertical space in typography. TeX treats characters with a [category code](#) of 10 as a space. By default this includes the normal space character and the tab character . The term “whitespace” may also include the [EOL](#) character.

SUMMARY OF COMMANDS AND ENVIRONMENTS

Further information about the commands or environments summarised here may be obtained from: the package or class documentation via `texdoc <name>` (for class or package commands or environments); *The TeXbook* [46] (for TeX primitives); the $\text{\LaTeX}2\epsilon$ source documentation via `texdoc source2e` (for \LaTeX Kernel commands or environments); the PDFTeX documentation via `texdoc pdftex` (for PDFTeX primitives); the ϵ -TeX documentation via `texdoc etex` (for ϵ -TeX primitives). If you use an up-to-date version of PDF \LaTeX you will be able to use all the primitives described here. Older versions of \LaTeX may not have some of the PDFTeX or ϵ -TeX primitives.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Symbols

aspect ratio. [§10.3]

!

!

Defined in: arara directive.

Defined in: xcolor package.

Logical not. [§1.2]

Used in colour specifications to indicate colour mixtures or percentages. [§12.1]

!

Defined in: graphicx package.

Used in `\resizebox` to maintain

Summary of Commands and Environments

`«`

A visual indication of a space in the code. When you type up the code, replace all instances of this symbol with a space via the space bar on your keyboard. [§1.0]

`#⟨digit⟩`

Defined in: L^AT_EX Kernel.

Replacement text for argument `⟨digit⟩`. (See Volume 1 [93, §8].) [§2.1]

`##⟨digit⟩`

Defined in: L^AT_EX Kernel.

Replacement text for argument `⟨digit⟩` when the command definition is included in the definition of another command. [§2.7]

`$`

Defined in: L^AT_EX Kernel.

Switches in and out of in-line math mode. (See Volume 1 [93, §9.1].) [§2.2]

`$`

Defined in: tikz calc library.

When used with the tikz calc library this is used to indicate co-ordinate calculations. [§10.3]

`%`

Defined in: L^AT_EX Kernel.

Comment character used to ignore everything up to and including the EOL character in the source code. This is often used to suppress unwanted space caused by the EOL in source code.

Sometimes comments are used to

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

provide information to applications that build your document, such as **arara**. See also [Volume 1 \[93, §2\]](#). [[§1.0](#)]

% arara:

Instruction to arara indicating how to build the document. This is ignored if you are not using arara. With v4.0 long directives may have line breaks provided the continuation lines start with

% arara: --> [[§1.2](#)]

&

Defined in: \LaTeX Kernel.

Alignment tab. [[§2.2](#)]

&

Defined in: arara directive.

Non-short-circuit logical and. [[§1.2](#)]

&&

Defined in: arara directive.

Logical and. [[§1.2](#)]

'

Defined in: \LaTeX Kernel.

Closing quote or apostrophe ' symbol in text mode or prime symbol ' in math mode. [[§4.3](#)]

''

Defined in: \LaTeX Kernel.

Closing double quote " symbol in text mode or double prime " in math mode. [[§2.7](#)]

--

Defined in: \LaTeX Kernel.

En-dash – symbol. (Normally used for number ranges.) [[§2.1](#)]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

--

Defined in: tikzpicture environment.

Used within the path specifications to indicate that a straight line should be drawn between two positions. [§12.1]

-->

Defined in: arara directive.

Continuation from previous line.
[§1.2]

Defined in: L^AT_EX Kernel.

Em-dash — symbol. (Normally used to indicate omissions or interruptions or to highlight a parenthetical element.) [§2.9]

<

Defined in: tikzpicture environment.

Start arrow tip [§12.1]

>{⟨decl⟩}

Defined in: array package.

Used in tabular-like environment column specifiers before l, r, c, p, m or b to insert ⟨decl⟩ directly in front of the entry for that column. [§4.3]

>

Defined in: tikzpicture environment.

End arrow tip [§12.1]

@{⟨text⟩}

Defined in: L^AT_EX Kernel.

Used in the argument of tabular-like environments to specify text to insert between columns. Since ⟨text⟩ replaces the usual inter-column space, this may also

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

be used with an empty argument to simply suppress that space.
[§10.3]

\@

Defined in: L^AT_EX Kernel.

Used when a sentence ends with a capital letter. This command should be placed after the letter and before the punctuation mark.

\@afterheading

Defined in: L^AT_EX Kernel.

Indicates a sectioning command has just been used, so the next paragraph should have its indentation suppressed. [§6.5]

\@auxout

Defined in: L^AT_EX Kernel.

Identifies the output stream for the document's auxiliary file. [§9.4]

\@currenvir

Defined in: L^AT_EX Kernel.

The name of the current environment. [§2.1]

\@enumdepth

Defined in: L^AT_EX Kernel.

T_EX count register that keeps track of the current `enumerate` level. [§6.5]

\@firstoftwo{\langle first \rangle}{\langle second \rangle}

Defined in: L^AT_EX Kernel ([internal command](#)).

Just does the first argument and discards the second argument. See also `\@secondoftwo`. [§11.1]

\@for{cs}:=\langle list \rangle \do{\langle body \rangle}

Defined in: L^AT_EX Kernel ([internal command](#)).

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Iterates through the [comma-separated list](#) and assigns the control sequence $\langle cs \rangle$ to the current item in the list so that it can be used as a placeholder in $\langle body \rangle$. The `xfor` package extends the functionality of this command, allowing you to terminate the loop at the end of the current iteration via `\@endfortrue`. [§[2.7](#)]

`\@ifundefined{\langle cs-name \rangle}{\langle true-part \rangle}{\langle false-part \rangle}`

Defined in: L^AT_EX Kernel ([internal command](#)).

Determines if the control sequence given by $\langle cs-name \rangle$ (without the leading backslash) is undefined (or `\relax`). [§[2.1](#)]

`\@namedef{\langle cs-name \rangle}{\langle arg-`

`syntax \rangle}{\langle definition \rangle}`

Defined in: L^AT_EX Kernel ([internal command](#)).

Defines the control sequence given by $\langle cs-name \rangle$ (without the leading backslash). [§[2.1](#)]

`\@nameuse{\langle cs-name \rangle}`

Defined in: L^AT_EX Kernel ([internal command](#)).

Uses the control sequence given by $\langle cs-name \rangle$ (without the leading backslash). [§[2.1](#)]

`\@secondoftwo{\langle first \rangle}{\langle second \rangle}`

Defined in: L^AT_EX Kernel ([internal command](#)).

Just does the second argument and discards the first argument. See also `\@firstoftwo`. [§[11.1](#)]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\@toodeep

Defined in: L^AT_EX Kernel.

Generates a “Too deeply nested” error. [§6.5]

[

Defined in: L^AT_EX Kernel.

Open delimiter of an optional argument. (See Volume 1 [93, §2.8.2].) [§1.0]

\&

Defined in: L^AT_EX Kernel.

Ampersand & symbol. [§2.2]

\

Defined in: L^AT_EX Kernel.

Escape character indicating a command. (See Volume 1 [93, §2.6].)

\\$

Defined in: L^AT_EX Kernel.

Dollar \$ symbol. [§2.2]

\#

Defined in: L^AT_EX Kernel.

Hash # symbol. [§2.2]

\%

Defined in: L^AT_EX Kernel.

Percent % symbol. [§2.2]

\{"c"\}

Defined in: L^AT_EX Kernel.

Umlaut over <c>. Example: \{"o\} produces ö. [§6.3]

\,

Defined in: L^AT_EX Kernel.

Thin space. [§4.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\[

Defined in: L^AT_EX Kernel.

Starts an unnumbered single-line of displayed maths. [§9.3]

\]

Defined in: L^AT_EX Kernel.

Starts a new row in environments that have a concept of rows rather than paragraphs (such as the tabular-style environments). This may have a starred version and/or an optional argument. [§2.7]

\'{⟨c⟩}

Defined in: L^AT_EX Kernel.

Acute accent over ⟨c⟩. Example:
\'{o} produces ó. [§5.1]

_

Defined in: L^AT_EX Kernel.

(Backslash followed by space character.) Horizontal spacing command. [§2.7]

\]

Defined in: L^AT_EX Kernel.

Ends an unnumbered single-line of displayed maths. [§9.3]

_

Defined in: L^AT_EX Kernel.

Underscore _ symbol. [§2.2]

\{

Defined in: L^AT_EX Kernel.

Left brace { character. In math mode may be used as a delimiter. [§2.2]

\}

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Right brace { character. In math mode may be used as a delimiter.
[§2.2]

]

Defined in: L^AT_EX Kernel.

Closing delimiter of an optional argument. (See Volume 1 [93, §2.8.2].) [§1.0]

$^{\{ \langle \text{maths} \rangle \}}$

Defined in: L^AT_EX Kernel (Math Mode).

Displays its argument as a superscript. [§2.2]

$_{\{ \langle \text{maths} \rangle \}}$

Defined in: L^AT_EX Kernel (Math Mode).

Displays its argument as a subscript. [§2.2]

'

Defined in: L^AT_EX Kernel.

Open quote ' symbol. [§4.3]

..

Defined in: L^AT_EX Kernel.

Open double quote " symbol. [§2.7]

{

Defined in: L^AT_EX Kernel.

Marks the beginning of a group. (See Volume 1 [93, §2.7].) [§1.0]

|

Defined in: arara directive.

Non-short-circuit logical or. [§1.2]

||

Defined in: arara directive.

Logical or. [§1.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

}

Defined in: L^AT_EX Kernel.

Marks the end of a group. (See
Volume 1 [93, §2.7].) [§1.0]

~

Defined in: L^AT_EX Kernel.

Unbreakable space. (See
Volume 1 [93, §4.3].) [§1.2]

A

\begin{about}

Defined in: pressrelease class.

For use within the `pressrelease` environment, this environment contains information about the company. [§6.2]

\accountdata

Defined in: isodoc class.

Generates a table containing the account information needed to pay the invoice. [§4.1]

\added[<options>]{<text>}

Defined in: changes package.

Indicates that the given text has been added. [§13.1]

\addplot[<path options>] <plot specs>;

Defined in: pgfplots package.

Adds a plot to the current image. [§12.5]

\addpoints

Defined in: exam class.

Enable the point-totalling commands. [§9.1]

\address{<text>}

Defined in: letter class.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Specifies the sender's address.
[§3.1]

`\addrfrom{<address>}`

Defined in: newlfm class.

Specifies the sender's address.
[§3.3]

`\addrto{<address>}`

Defined in: newlfm class.

Specifies the recipient's address.
[§3.3]

`\AddToBackground{<page-number>}{<picture-code>}`

Defined in: leaflet class (preamble only).

Indicates `picture` code to place on the page given by `<page-number>`.
The starred version refers to the sheet number instead. [§10.3]

`\addtolength{<register>}{<dimension>}`

Defined in: L^AT_EX Kernel.

Adds `<dimension>` to the value of the given length register. [§6.5]

`\advance<register> by <value>`

Defined in: T_EX primitive.

Increments the value stored in `<register>` by `<value>`. The `by` keyword may be omitted. [§2.1]

`\Alph{<counter>}`

Defined in: L^AT_EX Kernel.

Displays counter value as an upper case letter. (A, B, C, ..., Z) [§6.5]

`\alph{<counter>}`

Defined in: L^AT_EX Kernel.

Displays counter value as a lower case letter. (a, b, c, ..., z) [§6.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\and

Defined in: L^AT_EX Kernel.

Used to separate authors in

\author [§8.0]

\appenddynamiccontents{\{id\}}
\{text\}

Defined in: flowfram package.

Appends the given text to the contents of a dynamic frame.
[§10.5]

\appto{cs}{code}

Defined in: etoolbox package.

Appends `code` to the definition of the control sequence `cs`. Use \gappto for a global assignment.
[§2.1]

\arabic{counter}

Defined in: L^AT_EX Kernel.

Displays counter value as an Arabic number. (1, 2, 3, ...) [§6.5]

\begin{Argumentation}

Defined in: minutes package.

A list like environment to format an argument. [§6.3]

\AtBeginDocument{\code}

Defined in: L^AT_EX Kernel.

Specifies code that should be performed at the beginning of the document environment. This command has a cumulative effect.
[§7.3]

\author{\name}

Defined in: Most classes that have the concept of a title page.

Specifies the document author (or authors). This command doesn't

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

display any text so may be used in the preamble, but if it's not in the preamble it must be placed before `\maketitle`. Some classes, such as beamer, provide an optional argument for this command. [§2.3]

`\begin{axis}[\langle options \rangle]`

Defined in: pgfplots package.

Create a plot with normal Cartesian axes. [§12.5]

B

`\backgroundsetup{\langle options \rangle}`

Defined in: background package.

Sets the background options. [§6.4]

`\baselineskip`

Defined in: L^AT_EX Kernel.

A T_EX primitive that stores the minimum space from the bottom

of one line to the bottom of the next line in a paragraph. This is recalculated whenever the font changes. [§3.6]

`\bcbar[\langle options \rangle]{\langle value \rangle}`

Defined in: bchart package.

For use within the bchart environment, this draws a bar with the given value. [§12.3]

`\bcfontstyle`

Defined in: bchart package.

The font declaration used for the bar chart labels. [§12.3]

`\begin{bchart}[\langle options \rangle]`

Defined in: bchart package.

Creates a bar chart. [§12.3]

`\bclabel{\langle text \rangle}`

Defined in: bchart package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

For use within the `bchart` environment, this inserts a “free” label at the current position within the chart. [§12.3]

`\bcskip{\<length>}`

Defined in: `bchart` package.

For use within the `bchart` environment, this inserts a gap of the given length. [§12.3]

`\bc xlabel{\<text>}`

Defined in: `bchart` package.

For use within the `bchart` environment, this sets the x -axis label. [§12.3]

`\begin{\<env-name>}[\<env-option>]{\<env-arg-1>}...{\<env-arg-n>}`

Defined in: `LATEX Kernel`.

Starts an environment. (Must have a matching `\end`. See [Volume 1](#) [93, §2.15].) [§1.2]

`\bfseries`

Defined in: `LATEX Kernel`.

Switches to the bold weight in the current font family. See [Volume 1](#) [93, §4.5.1]. [§2.7]

`\BgThisPage`

Defined in: `background` package.

Indicates the background should be displayed on the current page. (For use with the `some pages` option.) [§6.4]

`\bibentry{\<key>}`

Defined in: `bibentry` package.

Prints the bibliographic entry for citation `\<key>`. [§5.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\bibitem[tag]{key}`

Defined in: L^AT_EX Kernel.

Indicates the start of a new reference in the bibliography. May only be used inside the contents of the `bibliography` environment [§5.2]

`\bibliography{bib-list}`

Defined in: L^AT_EX Kernel.

Inputs the `.bb1` file (if it exists) and identifies the name(s) of the bibliography database files where the citations are defined. [§1.2]

`\bibliographystyle{style-name}`

Defined in: L^AT_EX Kernel.

Specifies the bibliography style to be used by bibtex. [§1.2]

`\BigCircle`

Defined in: ifsym package with geometry option.

Produces a large open circle.

[§11.1]

`\bigskip`

Defined in: L^AT_EX Kernel.

Inserts a large vertical space. The size is given by the length `\bigskipamount`. [§10.3]

`\blackout{text}`

Defined in: censor package.

Redacts `<text>`, which can consist of one or more paragraphs. [§6.4]

`\begin{block}{title}`

Defined in: beamer class.

Puts its contents in a titled block. [§8.0]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\bonuspart[⟨points⟩]`

Defined in: exam class.

Like `\part` but indicates the points are bonus marks. [§9.1]

`\bonusquestion[⟨points⟩]`

Defined in: exam class.

Like `\question` but indicates the points are bonus marks. [§9.1]

`\bonussubpart[⟨points⟩]`

Defined in: exam class.

Like `\subpart` but indicates the points are bonus marks. [§9.1]

`\bonussubsubpart[⟨points⟩]`

Defined in: exam class.

Like `\subsubpart` but indicates the points are bonus marks. [§9.1]

`\bonussum`

Defined in: exsheets package.

Displays the total number of bonus points. (Requires two L^AT_EX runs to ensure it's up to date.) The starred version omits the unit. [§9.2]

`\bonustitledquestion{⟨title⟩}[⟨points⟩]`

Defined in: exam class.

Like `\titledquestion` but indicates the points are bonus marks. [§9.1]

`\boolean{⟨name⟩}`

Defined in: ifthen package.

May be used in the first argument of `\ifthenelse` to test the state of the named boolean variable. [§9.3]

`\boolfalse{⟨name⟩}`

Defined in: etoolbox package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Sets the given boolean variable's state to `false`. [§9.4]

`\booltrue{\name}`

Defined in: `etoolbox` package.

Sets the given boolean variable's state to `true`. [§9.4]

`\bottomrule[wd]`

Defined in: `booktabs` package.

Horizontal rule for the bottom of a `tabular` environment. [§2.6]

`\bracketedpoints`

Defined in: `exam` class.

Changes the points format to use square brackets instead of the default parentheses. [§9.1]

`\breakforeach`

Defined in: `pgffor` package.

When used inside the `<body>` part of `\foreach`, the loop will be terminated after the completion of the current iteration.

C

`\caption[<short-caption>]{<caption-text>}`

Defined in: `LATEX Kernel`.

Inserts the caption for a float such as a figure or table. [§2.6]

`\cc{\cc info}`

Defined in: Classes that define the `letter` environment.

Used to indicate the additional recipients of the letter. [§3.1]

`\cc{\names}`

Defined in: `minutes` package.

Specifies the distribution list. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\cclist{<text>}`

Defined in: newlfm class.

Sets distribution list. [§3.3]

`\censor{<text>}`

Defined in: censor package.

Redacts `<text>` by replacing the text with a filled black rectangle of the same size. [§6.4]

`\censor*{<size>}`

Defined in: censor package.

Alternative to `\censor` when the sensitive text should be omitted from the document source. [§6.4]

`\censorbox[<declarations>]{<contents>}`

Defined in: censor package.

Redacts `<contents>` (a box, such as a `tabular` environment) by replacing

the contents with a filled black rectangle of the same size. [§6.4]

`\censorbox*{<declarations>}{<width>}{<height>}{<depth>}`

Defined in: censor package.

Alternative to `\censorbox` when the sensitive text should be omitted from the document source. [§6.4]

`\begin{center}`

Defined in: L^AT_EX Kernel.

Centres its contents and places a small vertical gap above and below the environment. This gap can interfere with the vertical spacing in `figure` or `table` environments, so the use of `center` within a `float` is considered inappropriate. It can, however, be used outside a `float` where the gap may be useful to separate its

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

contents from the previous and following paragraphs. [§3.0]

`\centering`

Defined in: L^AT_EX Kernel.

Switches the paragraph alignment to centred. (See Volume 1 [93, §2.12].) [§2.6]

`\chair{\langle name \rangle}`

Defined in: meetingmins class.

Sets the name of the chair (for use within `\setmembers` and `\setpresent`). [§6.3]

`changed(\langle ref \rangle)`

Defined in: arara directive.

Evaluates to true if the given file has changed. The argument `\langle ref \rangle` may either be a string "`\langle extension \rangle`" which indicates the

file extension or a file reference

`toFile("filename").` [§1.2]

`\chapter[\langle short-title \rangle]{\langle title \rangle}`

Defined in: Book-style classes (such as `scrbook` or `scrreprt`) that have the concept of chapters.

Inserts a chapter heading. [§6.5]

`\CheckBox[\langle options \rangle]{\langle label \rangle}`

Defined in: hyperref package.

A check box (for use within the `Form` environment.) [§11.2]

`\begin{checkboxes}`

Defined in: exam class.

A list environment with checkbox choices as the items in the list. Items are specified via `\choice`. [§9.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\CheckBox

Defined in: wasysym package.

Produces a square with a tick in it
☒. [§11.1]

\choice

Defined in: exam class.

For use in one of the choices or checkboxes environments, this command starts a new choice. Use \CorrectChoice instead of \choice to indicate the correct choice. [§9.1]

\ChoiceMenu[*options*]{*label*}{*choices*}

Defined in: hyperref package.

A list menu, popup menu, combo menu or set of radio buttons (for use within the Form environment.) [§11.2]

\begin{choices}

Defined in: exam class.

A list environment with labelled choices as the items in the list. Items are specified via \choice. [§9.1]

\circle{*diameter*}

Defined in: LATEX Kernel.

For use in the argument of \put, this command draws a circle with the given diameter (specified in terms of \unitlength. The starred version fills the circle. [§10.1]

\cite[*text*]{*key list*}

Defined in: LATEX Kernel.

Inserts the citation markers of each reference identified in the key list. A second run is required to ensure the reference is correct.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Some packages redefine `\cite` to have two optional arguments. (See Volume 2 [96, §5].) [§1.2]

`\ClassError{<class-name>}{<error-message>}{<help-message>}`

Defined in: L^AT_EX Kernel.

Displays an error message for the given class. [§11.1]

`\clearpage`

Defined in: L^AT_EX Kernel.

Inserts a page break and processes any unprocessed floats.

`\closeline{<text>}`

Defined in: newlfm class.

The closing text. [§3.3]

`\closing{<sign-off text>}`

Defined in: Classes that define the `letter` environment.

Typesets the signing off text at the end of the letter. [§3.1]

`\Collect@Body{cs}`

Defined in: environ package.

As amsmath's `\collect@body` but allows paragraph breaks within the environment. [§11.1]

`\collect@body{cs}`

Defined in: amsmath package.

Collects the contents of the current environment and passes it to the command `(cs)` which should take an argument. The environment contents may not contain any paragraph breaks. [§11.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\begin{collectinmacro}{\macro}{\before}{\after}`

Defined in: collect package.

Collects the body of the environment and stores `\before\body\after` in the given macro. [§11.1]

`\color[<model>]{<specs>}`

Defined in: color and xcolor packages.

A declaration that switches the current foreground colour to the given specification. [§10.3]

`\colorbox[<model>]{<specs>}{<text>}`

Defined in: color package.

Produces a box containing `<text>` with the background given by

`<specs>` for the given colour model. [§10.5]

`\computeleftedgeeven{<cs>}`

Defined in: flowfram package.

Gets the co-ordinate of the left edge of the even numbered pages relative to the typeblock and stores it in the supplied control sequence. [§11.0]

`\computeleftedgeodd{<cs>}`

Defined in: flowfram package.

Gets the co-ordinate of the left edge of the odd numbered pages relative to the typeblock and stores it in the supplied control sequence. [§11.0]

`\Contra`

Defined in: minutes package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

For use within the [Argumentation](#) environment, this indicates the start of an important item against the argument. [§6.3]

`\contra`

Defined in: minutes package.

For use within the [Argumentation](#) environment, this indicates the start of an item against the argument. [§6.3]

`\CorrectChoice`

Defined in: exam class.

Used instead of `\choice` to indicate the correct choice. [§9.1]

`\correctitem`

Defined in: probsoln package.

For use within the `textenum`, this command may be used in place of

`\item` to indicate a correct choice. If the solutions aren't displayed this command behaves the same as `\item`. [§9.3]

`\correctitemformat{\langle marker \rangle}`

Defined in: probsoln package.

The format used by `\correctitem`. [§9.3]

`\cos`

Defined in: L^AT_EX Kernel (Math Mode).

Typesets cos function name. [§9.1]

`\begin{coverpages}`

Defined in: exam class.

Contains material to go before the start of the exam. [§9.1]

`\csappto{\langle cs-name \rangle}{\langle code \rangle}`

Defined in: etoolbox package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

As `\appto` but requires the name (without the leading backslash) of the control sequence. Use `\csgappto` for a global assignment. [§2.1]

```
\csdef{\cs-name}{arg-syntax}  
{definition}
```

Defined in: etoolbox package.

This is analogous to `\def` except that the name of the control sequence (without the initial backslash) is supplied. [§2.1]

```
\cseappo{\cs-name}{code}
```

Defined in: etoolbox package.

As `\eappto` but requires the name (without the leading backslash) of the control sequence. Use `\csxappto` for a global assignment. [§2.1]

```
\csedef{\cs-name}{arg-syntax}  
{definition}
```

Defined in: etoolbox package.

This is analogous to `\edef` except that the name of the control sequence (without the initial backslash) is supplied. [§2.1]

```
\csepreto{\cs-name}{code}
```

Defined in: etoolbox package.

As `\epreto` but requires the name (without the leading backslash) of the control sequence. Use

```
\csxpreto
```

 for a global assignment.
[§2.1]

```
\csgappto{\cs-name}{code}
```

Defined in: etoolbox package.

As `\gappto` but requires the name (without the leading backslash) of the control sequence. [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\csgdef{\langle cs-name\rangle}{\langle arg-syntax\rangle}{\langle definition\rangle}`

Defined in: etoolbox package.

This is analogous to `\gdef` except that the name of the control sequence (without the initial backslash) is supplied. [§2.1]

`\csgpreto{\langle cs-name\rangle}{\langle code\rangle}`

Defined in: etoolbox package.

Global version of `\cspreto`. [§2.1]

`\cslet{\langle cs-name\rangle}{\langle cs\rangle}`

Defined in: etoolbox package.

Analogous to `\let` except that the name of the control sequence (without the initial backslash) is supplied for the first argument. [§2.1]

`\csletcs{\langle new cs-name\rangle}{\langle org cs-name\rangle}`

Defined in: etoolbox package.

Analogous to `\let` except that the names of the control sequences (without the initial backslash) are supplied. [§2.1]

`\csname \langle cs-name\rangle\endcsname`

Defined in: TeX primitive.

Expands to the control sequence whose name (without the leading backslash) is given by `\langle cs-name\rangle`. If the control sequence isn't already defined, TeX will first define it to `\relax` before using it. [§2.1]

`\cspreto{\langle cs-name\rangle}{\langle code\rangle}`

Defined in: etoolbox package.

As `\preto` but requires the name (without the leading backslash) of

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

the control sequence. Use `\csgpreto` for a global assignment. [§2.1]

`\csuse{\langle cs-name \rangle}`

Defined in: etoolbox.

Executes the control sequence whose name (without the leading backslash) is given by `\langle cs-name \rangle`. If the command doesn't exist, this expands to an empty string. [§2.1]

`\csvloop[⟨auxiliary-commands⟩]{⟨list⟩}`

Defined in: etextools package.

Iterates through the [comma-separated list](#) using a handler macro provided in the optional argument. If none is provided, the command `\do` is used. The list may explicitly be a [comma-separated list](#) or it may

be a macro that expands to a list, unless the starred form is used. There are other variants to this command not described here. See the etextools documentation for further details. [§2.7]

`\csxappto{\langle cs-name \rangle}{⟨code⟩}`

Defined in: etoolbox package.

As `\xappto` but requires the name (without the leading backslash) of the control sequence. [§2.1]

`\csxdef{\langle cs-name \rangle}{⟨arg-syntax⟩}{⟨definition⟩}`

Defined in: etoolbox package.

This is analogous to `\xdef` except that the name of the control sequence (without the initial backslash) is supplied. [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
<code>\csxpreto{\langle cs-name\rangle}{\langle code\rangle}</code>	The body of the <code>CV</code> . [§5.1]	A N
Defined in: etoolbox package.		B O
Global version of <code>\csepreto</code> . [§2.1]		C P
<code>\CurrentOption</code>		D Q
Defined in: L ^A T _E X Kernel.		E R
Refers to the current option when used in the argument of <code>\DeclareOption*</code> [§11.1]	A headed list-like structure for use within the <code>cv</code> environment. [§5.1]	F S
<code>\CutLine{\langle page-number\rangle}</code>		G T
Defined in: leaflet class (preamble only).	Defined in: currvita package.	H U
Indicates that a cut line should be drawn to the left of the given page number. The starred version only draws a dotted line. The unstarred version draws a dotted line and a pair of scissors. [§10.3]	Specifies the location where the <code>CV</code> was written. [§5.1]	I V
		J W
		K X
		L Y
		M Z
	<code>\dashbox{\langle dash-length\rangle}{\langle w\rangle,\langle h\rangle}[{\langle align\rangle}]{\langle text\rangle}</code>	
	Defined in: picture environment.	
	Similar to <code>\framebox</code> but produces a dashed frame. [§10.1]	
	<code>\date{\langle text\rangle}</code>	
	Defined in: Most classes that have the concept of a title page.	

Summary of Commands and Environments

Specifies the document date. This command doesn't display any text so may be used in the preamble, but if it's not in the preamble it must be placed before `\maketitle`. If omitted, most classes assume the current date. Some classes, such as beamer, provide an optional argument for this command. [§5.1]

`\dateset{\langle date \rangle}`

Defined in: newlfm class.

Sets the date. [§3.3]

`\day`

Defined in: TeX primitive.

The current day of the month.
[§7.2]

`\DBIBcitekey`

Defined in: databib package.

For use with `\DTLforeachbibentry` or `\gDTLforeachbibentry`, this expands to the name of the current cite key. [§5.2]

`\DBIBentrytype`

Defined in: databib package.

For use with `\DTLforeachbibentry` or `\gDTLforeachbibentry`, this expands to the name of the current entry type (for example, book). [§5.2]

`\decision{\langle theme \rangle}{\langle short-description \rangle}{\langle long-description \rangle}`

Defined in: minutes package.

Specifies a decision, which is added to the list of decisions. [§6.3]

`\decision*{\langle short-description \rangle}{\langle long-description \rangle}`

Defined in: minutes package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Specifies a decision, which isn't added to the list of decisions. [§6.3]

`\decisiontheme{<theme>}{<title>}`

Defined in: minutes package.

Defines a decision theme which will be added to the list of decisions. [§6.3]

`\DeclareOption{<option>}{<code>}`

Defined in: L^AT_EX Kernel.

Declares an option for a class or package. [§11.1]

`\DeclareOption*{<code>}`

Defined in: L^AT_EX Kernel.

Declares the code for an unknown option. You can refer to the option within `<code>` using

`\CurrentOption`. [§11.1]

`\def{<cs>}{{arg-syntax}}{<definition>}`

Defined in: T_EX primitive.

This locally defines the command `<cs>` that has the given syntax. Use `\gdef` for global definitions. [§2.1]

`\DefaultHeightofCheckBox`

Defined in: hyperref package.

The default height of check boxes. [§11.2]

`\DefaultHeightofChoiceMenu`

Defined in: hyperref package.

The default height of choice boxes. [§11.2]

`\DefaultHeightofReset`

Defined in: hyperref package.

The default height of the reset button. [§11.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\DefaultHeightofSubmit

Defined in: hyperref package.

The default height of the submit button. [§11.2]

\DefaultHeightofText

Defined in: hyperref package.

The default height of single-lined text fields. [§11.2]

\DefaultHeightofTextMultiline

Defined in: hyperref package.

The default height of multi-lined text fields. [§11.2]

\DefaultWidthofCheckBox

Defined in: hyperref package.

The default width of check boxes. [§11.2]

\DefaultWidthofChoiceMenu

Defined in: hyperref package.

The default width of choice boxes. [§11.2]

\DefaultWidthofReset

Defined in: hyperref package.

The default width of the reset button. [§11.2]

\DefaultWidthofSubmit

Defined in: hyperref package.

The default width of the submit button. [§11.2]

\DefaultWidthofText

Defined in: hyperref package.

The default width of text fields. [§11.2]

Symbols

A N

B O

C P

D Q

E R

F S

G T

H U

I V

J W

K X

L Y

M Z

`\definechangesauthor[options]{id}`

Defined in: changes package.

Defines a tracked author. [§13.1]

`\begin{defproblem}[n]
[default-args]{label}
[option]`

Defined in: probsoln package.

Defines a new problem. (The contents may include the `onlysolution` environment for the solution.) [§9.3]

`\deleted[options]{text}`

Defined in: changes package.

Indicates that the given text has been deleted. [§13.1]

`\Delta`

Defined in: L^AT_EX Kernel (Math Mode).

Greek upper case delta Δ . [§9.3]

`\descfont`

Defined in: leaflet class.

The font declaration used by the item labels in the `description` environment. [§10.3]

`\begin{description}`

Defined in: Most class files.

Labelled list. [§10.3]

`\dimexprdimension expression`

Defined in: ϵ -T_EX primitive.

Expands to the value given by the dimension expression. [§2.1]

`\ding{n}`

Defined in: pifont package.

Inserts PostScript ZapfDingbats character with code *n*, which must be an integer. [§2.9]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\begin{dinglist}{<number>}`

Defined in: pifont package.

A list where the item marker is given by character `<number>` in the Zapf Dingbats font. [§10.5]

`\Discount{<description>}{<amount>}`

Defined in: invoice package.

For use within the `invoice` environment, this command is used to specify a discount. [§4.2]

`\divide<register> by <value>`

Defined in: TeX primitive.

Divides the value stored in `<register>` by `<value>`. The `by` keyword may be omitted. [§2.1]

`\do{<item>}`

Defined in: etoolbox package.

Handler macro used by commands like `\docslist` and `\csvloop`. The argument is the item in the current iteration of the list. [§2.7]

`\docslist{<item1,item2,...>}`

Defined in: etoolbox package.

Loops over the given comma-separated list and executes the command `\do` for every item in the list, using the item as the argument of `\do`. It's up to the user to define `\do` as appropriate. [§2.7]

`\begin{document}`

Defined in: L^AT_EX Kernel.

The body of the document. [§6.3]

`\documentclass[<option-list>]`

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`{<class-name>}`

Defined in: L^AT_EX Kernel.

Loads the document class file, which sets up the type of document you wish to write. (See **Volume 1** [93, §4].) [§1.0]

`\doforrandN{<n>}{{<cs>}}{<list>}{<body>}`

Defined in: probsoln package.

Iterates over a randomly selected $\langle n \rangle$ items in a **comma-separated list**. At each iteration, $\langle cs \rangle$ is defined to be the currently selected item and $\langle body \rangle$ is performed. [§9.3]

`\dolistcsloop{<list-csname>}`

Defined in: etoolbox package.

Similar to `\dolistloop` except the name (without the preceding

backslash) of the list control sequence is used. [§2.7]

`\dolistloop{<list-cs>}`

Defined in: etoolbox package.

Iterates over all items in the list macro $\langle list-cs \rangle$ and performs `\do{<item>}` for each item. [§2.7]

`\dotfill`

Defined in: L^AT_EX Kernel.

Fills the remaining space with a dotted line. [§10.3]

`\draw <specification>;`

Defined in: tikz package.

For use within the `tikzpicture` environment. Draws the path with the given specification using the current path stroking operations. [§12.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\dropoints

Defined in: exam class.

Prints the question points. This command should only occur at the end of a paragraph or between paragraphs. (Used with \pointsdroppedatright.) [§9.1]

\DTLabs{\cs}{\number}

Defined in: datatool package.

Computes the absolute value of $\langle number \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number should be formatted according to the current locale (set via \DTLsetnumberchars) and may optionally be prefixed with a known currency identifier. See the datatool documentation for further details. [§2.1]

\dtlabs{\cs}{\number}

Defined in: datatool package.

Computes the absolute value of $\langle number \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]

\DTLadd{\cs}{\number1}{\number2}

Defined in: datatool package.

Adds $\langle number1 \rangle$ to $\langle number2 \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number should be formatted according to the current locale (set via \DTLsetnumberchars) and may optionally be prefixed with a

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

known currency identifier. See the datatool documentation for further details. [§2.1]

```
\dtladd{<cs>}{{<number1>}{<number2>}}
```

Defined in: datatool package.

Adds *<number1>* to *<number2>* and stores the result in *<cs>*, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]

```
\DTLassign{<db-name>}{{<row-idx>}}{<assign-list>}
```

Defined in: datatool package.

Applies the assignment list, which has the same format as for `\DTLforeach` and `\DTLforeach*`, to

the given row. Row indexes start from 1. [§2.8]

```
\DTLassignfirstmatch{<db-name>}{{<col-label>}}{<value>}{<assign-list>}
```

Defined in: datatool package.

Finds the first row in the database *<db-name>* where entry in the column identified by the label *<col-label>* matches *<value>* and applies the assignment list, which has the same format as for `\DTLforeach` and `\DTLforeach*`.

Note that no expansion is performed on *<value>*. See also `\xDTLassignfirstmatch`. [§2.8]

```
\DTLbaratbegintikz
```

Defined in: databar package.

Hook used by `\DTLbarchart` or `\DTLmultibarchart` at the start of

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

the `tikzpicture` environment. [§12.3]

`\DTLbaratendtikz`

Defined in: databar package.

Hook used by `\DTLbarchart` or `\DTLmultibarchart` at the start of the `tikzpicture` environment. [§12.3]

`\DTLbarchart[<condition>]{<settings>}(<db-name>){<assign-list>}`

Defined in: databar package.

Generates a bar chart from a column of the datatool database called `<db-name>`. [§12.3]

`\DTLbarchartlength`

Defined in: databar package.

A length register storing the total length of the y -axis. [§12.3]

`\DTLbarlabeloffset`

Defined in: databar package.

A length register storing the offset between the x -axis and the lower bar label. [§12.3]

`\DTLbaroutlinecolor`

Defined in: databar package.

A macro that expands to the colour used to draw the bar outlines. [§12.3]

`\DTLbaroutlinewidth`

Defined in: databar package.

A length register that stores the width of the bar outline. A zero width indicates that the outline shouldn't be drawn. [§12.3]

`\DTLbarwidth`

Defined in: databar package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

A length register storing the width of each bar. [§12.3]

`\DTLbibfield{<field-name>}`

Defined in: databib package.

(For use within the final argument of `\DTLforeachbibentry` or `\gDTLforeachbibentry`.) This command displays the value of entry in the column whose label is given by `<field-name>`. [§5.2]

`\DTLbibfieldlet{<cs>}{{<field-name>}}`

Defined in: databib package.

(For use within the final argument of `\DTLforeachbibentry` or `\gDTLforeachbibentry`.) This command assigns the value of entry in the column whose label is given by `<field-name>` to the control sequence `<cs>`. [§5.2]

`\dtlbreak`

Defined in: datatool package.

When used in the body of commands like `\DTLforeach`, this signifies that the loop should be terminated at the end of the current iteration. [§2.7]

`\dtlcompare{<register>}{{<text1>}}{<text2>}`

Defined in: datatool package.

A case-sensitive comparison handler for use with `\dtlsort`. [§2.4]

`\DTLcustombibitem{<marker-code>}{{<ref-text>}}{<key>}`

Defined in: databib package.

This command is analogous to `\bibitem[<label>]{<key>}` except it

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

uses `<marker code>` instead of
`\item[<label>]`. [§5.2]

`\dtldefaultkey`

Defined in: datatool package.

The prefix to use when generating
a default column label. [§2.2]

`\dtldisplayafterhead`

Defined in: datatool package.

Hook after the header row in
`\DTLdisplaydb` and

`\DTLdisplaylongdb`. [§2.6]

`\DTLdisplaydb[<omit-list>]{<db-name>}`

Defined in: datatool package.

Displays the database identified by
`<db-name>` in a `tabular`
environment. [§2.2]

`\dtldisplayendtab`

Defined in: datatool package.

Hook used at the end of

`\DTLdisplaydb` and

`\DTLdisplaylongdb`. [§2.6]

`\DTLdisplaylongdb[<options>]{<db-name>}`

Defined in: datatool package.

Displays the database identified by
`<db-name>` in a `longtable`
environment. [§2.6]

`\dtldisplaystarttab`

Defined in: datatool package.

Hook used at the start of

`\DTLdisplaydb` and

`\DTLdisplaylongdb`. [§2.6]

`\DTLdiv{<cs>}{<number1>}{<number2>}`

Defined in: datatool package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Computes $\langle number1 \rangle$ divided by $\langle number2 \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number should be formatted according to the current locale (set via `\DTLsetnumberchars`) and may optionally be prefixed with a known currency identifier. See the datatool documentation for further details. [§2.1]

```
\dtldiv{\{cs\}}{\{number1\}}  
{\{number2\}}
```

Defined in: datatool package.

Computes $\langle number1 \rangle$ divided by $\langle number2 \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix).

[§2.1]

```
\DTLdobarcolor{\{n\}}
```

Defined in: databar package.

Sets the current text colour to the colour of the $\langle n \rangle$ th bar. [§12.3]

```
\DTLcurrentpiesegmentcolor
```

Defined in: datapie package.

As `\DTLdopiesegmentcolor` but for the current segment. [§12.2]

```
\DTLdopiesegmentcolor{\{n\}}
```

Defined in: datapie package.

Switches the current text colour to that of the $\langle n \rangle$ th segment. [§12.2]

```
\DTLendpt
```

Defined in: databar package.

For use within the definition of `\DTLeverybarhook`, this can be

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

used to reference the end of the bar. [§12.3]

\DTLeverybarhook

Defined in: databar package.

Hook used by \DTLbarchart or \DTLmultibarchart at each bar. [§12.3]

\dtlexpandnewvalue

Defined in: datatool package.

Switches on the value expansion when using \DTLnewdbentry [§9.4]

\DTLforeach[*condition*]{*db-name*}{*assign-list*}{*body*}

Defined in: datatool package.

Iterates through each row of the database identified by *db-name* and does *body* if *condition* is met for that row. [§2.6]

\DTLforeach*[*condition*]{*db-name*}{*assign-list*}{*body*}

Defined in: datatool package.

A read-only version of \DTLforeach. If no modifications need to be made to the database, this is the better version to use as it's quicker. (How much quicker depends on the size of the database.) [§2.7]

\DTLforeachbibentry
[*condition*]{*db-name*}{*body*}

Defined in: databib package.

Iterates through the database (loaded by \DTLloadbbl) called *db-name*, and performs *body* on each row where *condition* is met. The starred version is read-only. [§5.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
<code>\DTLformatbibentry</code>	Global version of <code>\DTLabs</code> [§2.1]	A N
Defined in: databib package. (For use within the final argument of <code>\DTLforeachbibentry</code> or <code>\gDTLforeachbibentry</code> .) This command displays the current row according to the format for its given entry type. [§5.2]	<code>\DTLgadd{\{cs\}}{\{number1\}}{\{number2\}}</code> Defined in: datatool package. Global version of <code>\DTLadd</code> [§2.1]	B O C P D Q E R F S
<code>\DTLformatthisbibentry{\{db-name\}}{\{key\}}</code>	<code>\DTLgdiv{\{cs\}}{\{number1\}}{\{number2\}}</code> Defined in: datatool package. Global version of <code>\DTLdiv</code> [§2.1]	G T H U I V J W K X
Defined in: databib package. Similar to <code>\DTLformatbibentry</code> but can be used outside <code>\DTLforeachbibentry</code> / <code>\gDTLforeachbibentry</code> to format the referenced identified by <code>\langle key \rangle</code> in the database <code>\langle db-name \rangle</code> . [§5.2]	<code>\DTLgmul{\{cs\}}{\{number1\}}{\{number2\}}</code> Defined in: datatool package. Global version of <code>\DTLmul</code> [§2.1]	L Y M Z
<code>\DTLgabs{\{cs\}}{\{number\}}</code>	<code>\DTLgneg{\{cs\}}{\{number\}}</code> Defined in: datatool package. Global version of <code>\DTLneg</code> [§2.1]	

Summary of Commands and Environments

\DTLground{\{cs\}}{\{number\}}
{\{num-digits\}}

Defined in: datatool package.

Global version of \DTLround [§2.1]

\DTLgsub{\{cs\}}{\{number1\}}
{\{number2\}}

Defined in: datatool package.

Global version of \DTLsub [§2.1]

\dtlicompare{\{register\}}{\{text1\}}
{\{text2\}}

Defined in: datatool package.

A case-insensitive comparison
handler for use with \dtlsort.
[§2.4]

\DTLifbibfieldexists{\{field-
name\}}{\{true-part\}}{\{false-
part\}}

Defined in: databib package.

(For use within the final argument
of \DTLforeachbibentry or
\gDTLforeachbibentry.) Does
{*true-part*} if the entry in the
column whose label is given by
{*field-name*} is non-null.
Otherwise does {*false-part*}. [§5.2]

\DTLifdbempty{\{db-name\}}{\{true-
part\}}{\{false-part\}}

Defined in: datatool package.

Checks if the datatool internal
database called {*db-name*} is
empty. [§5.2]

\DTLiffirstrow{\{true\}}{\{false\}}

Defined in: datatool package.

Provided for use in the {*body*}
part of \DTLforeach this does
{*true*} if it's on the iteration is on
the first row otherwise it does
{*false*}. [§2.7]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\DTLifinlist{\item}{\list}
{\true}{\false}

Defined in: datatool package.

Checks if the given item is in the given comma-separated list.

A one-level expansion is performed on *list* but not on *item*. [§2.7]

\DTLiflastrow{\true}{\false}

Defined in: datatool package.

Provided for use in the *body* part of \DTLforeach this does *true* if it's on the iteration is on the last row otherwise it does *false*. [§2.7]

\DTLifnull{\cs}{\true}{\false}

Defined in: datatool package.

If the control sequence *cs* is null (as defined by datatool) this does

\DTLifnullorempty{\cs}{\true}{\false}
[§2.9]

\DTLifnullorempty{\cs}{\true}{\false}

Defined in: datatool package.

This is a combination of \DTLifnull and \ifdefempty. If the control sequence *cs* is null (as defined by datatool) or empty this does *true* otherwise this does *false*. [§2.9]

\dtlifnumeq{\number1}{\number2}{\true}{\false}

Defined in: datatool-base package.

Checks if *number1* equals *number2* and does *true* if true, otherwise does *false*. The numbers may be integers or decimals [§9.4]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\dtllastloadeddb

Defined in: datatool package.

When you load a .dbtex file, this command is set to the label associated with the data in that file. [§2.2]

\dtlletterindexcompare
{register}{text1}{text2}

Defined in: datatool package.

A letter-ordering comparison handler for use with \dtlsort. [§2.4]

\DTLloadbb1[⟨bbl⟩]{⟨db-name⟩}
{⟨bib-list⟩}

Defined in: databib package.

Loads bibliography data from the .bbl file given by ⟨bbl⟩ and stores it in the datatool internal database called ⟨db-name⟩. This also sets

the bibliography style to databib.bst and identifies the list of .bib files where the bibliography data is defined. [§5.2]

\DTLloaddb[⟨options⟩]{⟨db-name⟩}{⟨filename⟩}

Defined in: datatool package.

Loads the data given in the CSV file ⟨filename⟩ and stores it in a datatool database called ⟨db-name⟩. [§2.2]

\DTLloaddbtex{⟨cs⟩}{⟨filename⟩}

Defined in: datatool package.

Inputs the and assigns the database name to the command ⟨cs⟩ [§2.2]

\DTLloadrawdb[⟨options⟩]{⟨db-name⟩}{⟨filename⟩}

Defined in: datatool package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

As `\DTLloaddb` but maps nine of the ten special characters to L^AT_EX commands that display the relevant symbol. [§2.2]

`\DTLmaketabspace`

Defined in: datatool package.

Changes the `category code` of the tab separator to 10 (space). [§2.2]

`\DTLmidpt`

Defined in: databar package.

For use within the definition of `\DTLeverybarhook`, this can be used to reference the mid point of the bar. [§12.3]

`\DTLmonthname{\langle number \rangle}`

Defined in: databib package.

Displays the month name for the month given by `\langle number \rangle`. [§5.2]

`\DTLmul{\langle cs \rangle}{\langle number1 \rangle}{\langle number2 \rangle}`

Defined in: datatool package.

Computes `\langle number1 \rangle` multiplied by `\langle number2 \rangle` and stores the result in `\langle cs \rangle`, which must be a control sequence. The number should be formatted according to the current locale (set via

`\DTLsetnumberchars`) and may optionally be prefixed with a known currency identifier. See the datatool documentation for further details. [§2.1]

`\dtlmul{\langle cs \rangle}{\langle number1 \rangle}{\langle number2 \rangle}`

Defined in: datatool package.

Computes `\langle number1 \rangle` multiplied by `\langle number2 \rangle` and stores the result in `\langle cs \rangle`, which must be a

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]

`\DTLmultibarchart{<condition>} {<settings>} {<db-name>} {<assign-list>}`

Defined in: databar package.

Generates a bar chart with grouped data from columns of the datatool database called `<db-name>`. [§12.3]

`\DTLneg{<cs>} {<number>}`

Defined in: datatool package.

Negates `<number>` and stores the result in `<cs>`, which must be a control sequence. The number should be formatted according to the current locale (set via

`\DTLsetnumberchars`) and may optionally be prefixed with a known currency identifier. See the datatool documentation for further details. [§2.1]

`\dtlneg{<cs>} {<number>}`

Defined in: datatool package.

Negates `<number>` and stores the result in `<cs>`, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]

`\DTLnewdb{<db-name>}`

Defined in: datatool package.

Creates a new database called `<db-name>`. [§9.4]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\DTLnewdbentry{\langle db-name \rangle}{\langle col-label \rangle}{\langle value \rangle}

Defined in: datatool package.

Adds an entry to the current row of the named database. [§9.4]

\DTLnewrow{\langle db-name \rangle}

Defined in: datatool package.

Adds a new row to the database. This new row becomes the current row when adding new entries. [§9.4]

\DTLnumitemsinlist{\langle list \rangle}{\langle cs \rangle}

Defined in: datatool package.

Counts the number of non-empty items in the given **comma-separated list** and stores the result in $\langle cs \rangle$. A one-level expansion is performed on $\langle list \rangle$. [§2.7]

\DTLpieatbegintikz

Defined in: datapie package.

Hook performed at the start of \DTLpiechart. [§12.2]

\DTLpieatendtikz

Defined in: datapie package.

Hook performed at the end of \DTLpiechart. [§12.2]

\DTLpiechart[\langle condition \rangle]

{\langle settings \rangle}{\langle db-name \rangle}{\langle assign-list \rangle}

Defined in: datapie package.

Creates a pie chart from the data given in the database $\langle db-name \rangle$. [§12.2]

\DTLpiepercent

Defined in: datapie package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

May be used in the inner or outer label to access the percentage value of the pie chart variable. [§12.2]

\DTLpievariable

Defined in: datapie package.

May be used in the inner or outer label to access the value of the pie chart variable. [§12.2]

\DTLprotectedsaverawdb{*dbname*}{{*filename*}}

Defined in: datatool package.

Like \DTLsaverawdb but works with databases that contain fragile commands. [§9.4]

\DTLradius

Defined in: datapie package.

The pie chart radius [§12.2]

\DTLrawmap{*string*}
 {*replacement*}

Defined in: datatool package.

Defines extra mappings for \DTLloadrawdb. [§2.2]

\DTLround{*cs*}{{*number*}}
 {*num-digits*}

Defined in: datatool package.

Rounds *number* to *num-digits* decimal places and stores the result in *cs*, which must be a control sequence. The number should be formatted according to the current locale (set via \DTLsetnumberchars) and may optionally be prefixed with a known currency identifier. See the datatool documentation for further details. [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\dtlround{\langle cs \rangle}{\langle number \rangle}{\langle num-digits \rangle}`

Defined in: datatool package.

Rounds `\langle number \rangle` to `\langle num-digits \rangle` decimal places and stores the result in `\langle cs \rangle`, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]

`\DTLsaferawdb{\langle db-name \rangle}{\langle filename \rangle}`

Defined in: datatool package.

Saves the database in the format that can be loaded by

`\DTLloaddbtex` and the `datatooltk` application. If the database contains any fragile commands, use `\DTLprotectedsaferawdb` instead. [§9.4]

`\DTLsetbarcolor{\langle n \rangle}{\langle colour \rangle}`

Defined in: databar package.

Sets the `\langle n \rangle`th bar colour to `\langle colour \rangle`. [§12.3]

`\DTLsetdelimiter{\langle character \rangle}`

Defined in: datatool package.

Specifies the delimiter character for CSV files. [§2.2]

`\DTLsetheader{\langle db-name \rangle}{\langle col-label \rangle}{\langle header \rangle}`

Defined in: datatool package.

Assigns a header for the column identified by `\langle col-label \rangle` in the database labelled `\langle db-name \rangle`. [§2.2]

`\DTLsetnumberchars{\langle number group character \rangle}{\langle decimal character \rangle}`

Defined in: datatool package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Sets the number group and decimal character for real numbers. See the `datatool` package for further details. [§2.1]

`\DTLsetpiesegmentcolor{\langle n \rangle}{\langle colour \rangle}`

Defined in: `datapie` package.

Sets the colour for the $\langle n \rangle$ th pie chart segment. [§12.2]

`\DTLsetseparator{\langle character \rangle}`

Defined in: `datatool` package.

Specifies the separator character for `CSV` files. [§2.2]

`\DTLsettabseparator`

Defined in: `datatool` package.

Sets the separator character for `CSV` files to the tab character. [§2.2]

`\DTLsort[\langle replacement \rangle]{\langle criteria \rangle}{\langle db-name \rangle}`

Defined in: `datatool` package.

A shortcut for `\dtlsort` where the comparison handler is `\dtlcompare` (case-sensitive). [§2.4]

`\dtlsort[\langle replacement \rangle]{\langle criteria \rangle}{\langle db-name \rangle}{\langle handler \rangle}`

Defined in: `datatool` package.

Sorts the data identified by $\langle db-name \rangle$ according to the columns listed in the $\langle criteria \rangle$ using the given comparison handler control sequence. [§2.4]

`\DTLsort*[\langle replacement \rangle]{\langle criteria \rangle}{\langle db-name \rangle}`

Defined in: `datatool` package.

A shortcut for `\dtlsort` where the comparison handler is

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
\dtlicompare	(case-insensitive). [§2.4]	A N
\DTLstartpt		B O
Defined in: databar package.		C P
For use within the definition of \DTLeverybarhook, this can be used to reference the start of the bar. [§12.3]		D Q
\DTLsub{\cs}{\number1} {\number2}		E R
Defined in: datatool package.	\dtlsub{\cs}{\number1} {\number2}	F S
Computes $\langle number1 \rangle$ minus $\langle number2 \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number should be formatted according to the current locale (set via \DTLsetnumberchars) and may optionally be prefixed with a known currency identifier. See the	Defined in: datatool package. Computes $\langle number1 \rangle$ minus $\langle number2 \rangle$ and stores the result in $\langle cs \rangle$, which must be a control sequence. The number must be a plain decimal number (a full stop as a decimal point, no number grouping and no currency prefix). [§2.1]	G T
	\dtlwordindexcompare{\register} {\text1}{\text2}	H U
	Defined in: datatool package.	I V
	A word-ordering comparison handler for use with \dtlsort. [§2.4]	J W
		K X
		L Y
		M Z

Summary of Commands and Environments

`\DTMdate{<date>}`

Defined in: `datetime2` package.

Displays the given date according to the current date format. [§7.1]

`\DTMnow`

Defined in: `datetime2` package.

Inserts into the output file the date and time when the `LATEX` application created it from the source code. [§7.1]

`\DTMsavedate{<name>}{{<date>}}`

Defined in: `datetime2` package.

Stores the given date. [§7.1]

`\DTMsavetimestamp{<name>}{{<data>}}`

Defined in: `datetime2` package.

Stores the given date and time data. [§7.1]

`\DTMuse{<name>}`

Defined in: `datetime2` package.

Displays the previously saved date and time stamp. [§7.1]

`\DTMusedate{<name>}`

Defined in: `datetime2` package.

Displays the previously saved date. [§7.1]

`\DTMusetime{<name>}`

Defined in: `datetime2` package.

Displays the previously saved time. [§7.1]

`\begin{dynamiccontents}{<id>}`

Defined in: `flowfram` package.

Sets the contents of a dynamic frame. (Verbatim not allowed.) [§10.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
E	\ecvaddress{\<address\>}	A N
\eappto{cs}{\<code\>}	Defined in: etoolbox package.	B O
Similar to \appto but expands \<code>. Use \xappto for a global assignment. [§2.1]	Specifies your address. [§5.2]	C P
\EBC{\<description\>}{\<amount\>}	\ecvafterpicture{\<text\>}	D Q
Defined in: invoice package.	Defined in: europecv class.	E R
For use within the <code>invoice</code> environment, this command is used to specify a local expense. [§4.2]	Specifies text to include after insert your personal image. You can use \ecvspace within \<text>. [§5.2]	F S
\EBCi{\<description\>}{\<amount\>}	\ecvAOne	G T
Defined in: invoice package.	Defined in: europecv class.	H U
For use within the <code>invoice</code> environment, this command is like \EBC but although the amount is added to the total expense it's not itemized. [§4.2]	Shortcut for \ecvCEF{A1}{basic user}. [§5.2]	I V
	\ecvATwo	J W
	Defined in: europecv class.	K X
	Shortcut for \ecvCEF{A2}{basic user}. [§5.2]	L Y
		M Z

Summary of Commands and Environments

`\ecvbeforepicture{<text>}`

Defined in: europecv class.

Specifies text to include before insert your personal image. You can use `\ecvspace` within `<text>`. [§5.2]

`\ecvBOne`

Defined in: europecv class.

Shortcut for

`\ecvCEF{B1}{intermediate user}.`
[§5.2]

`\ecvBTwo`

Defined in: europecv class.

Shortcut for

`\ecvCEF{B2}{intermediate user}.`
[§5.2]

`\ecvCEF{<level>}{<descr>}`

Defined in: europecv class.

For use in the `<l1>`, ..., `<l5>` arguments of `\ecvlanguage` or `\ecvlastlanguage`. [§5.2]

`\ecvColSep{<width>}`

Defined in: europecv class.

Sets the column separation. [§5.2]

`\ecvCOne`

Defined in: europecv class.

Shortcut for

`\ecvCEF{C1}{proficient user}.`
[§5.2]

`\ecvCTwo`

Defined in: europecv class.

Shortcut for

`\ecvCEF{C2}{proficient user}.`
[§5.2]

`\ecvdateofbirth{<date>}`

Defined in: europecv class.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Specifies your date of birth. [§5.2]	<code>\ecvitem[<vspace>]{<left text>}{<right text>}</code>	Symbols
<code>\ecvemail{<address>}</code>	Defined in: europecv class.	A N
Specifies your email address. [§5.2]	Defines <code>\ecvitem</code> to add <code><left text></code> to the left of the vertical rule and <code><right text></code> to the right of the vertical rule. [§5.2]	B O
<code>\ecvfax{<fax number>}</code>	Defined in: europecv class.	C P
Specifies your fax number. [§5.2]	Typesets a row in the language table. Each of the <code><l1></code> , ..., <code><l5></code> arguments should be in the form <code>\ecvCEF{<level>}{<desc>}</code> . Use <code>\ecvlastlanguage</code> for the last row. [§5.2]	D Q
<code>\ecvfootername{<name>}</code>	Defined in: europecv class.	E R
Specifies your name as it will appear in the footer. If omitted, the name will be taken from that specified by <code>\ecvname</code> . [§5.2]	Typesets a row in the language table. Each of the <code><l1></code> , ..., <code><l5></code> arguments should be in the form <code>\ecvCEF{<level>}{<desc>}</code> . Use <code>\ecvlastlanguage</code> for the last row. [§5.2]	F S
<code>\ecvgender{<gender>}</code>	Defined in: europecv class.	G T
Specifies your gender. [§5.2]	Defines <code>\ecvlanguage</code> to add <code><symbol></code> to the left of the vertical rule and <code><language></code> to the right of the vertical rule. [§5.2]	H U
	Defined in: europecv class.	I V
	Defines <code>\ecvlanguagefooter</code> to add <code><symbol></code> to the left of the vertical rule and <code><language></code> to the right of the vertical rule. [§5.2]	J W
	Defined in: europecv class.	K X
	Defines <code>\ecvlastlanguage</code> to add <code><symbol></code> to the left of the vertical rule and <code><language></code> to the right of the vertical rule. [§5.2]	L Y
	Defined in: europecv class.	M Z

Summary of Commands and Environments

Typesets the footer of the language table and identifies the symbol to use as a footnote symbol, which should be the same as that used in

`\ecvlanguageheader.` [§5.2]

`\ecvlanguageheader{\symbol}`

Defined in: europecv class.

Typesets the header of the language table and identifies the symbol to use as a footnote symbol. [§5.2]

`\ecvlastlanguage[{\vspace}]{\language}{\l1}{\l2}{\l3}{\l4}{\l5}`

Defined in: europecv class.

Typesets the last row in the language table. Each of the $\langle l1 \rangle, \dots, \langle l5 \rangle$ arguments should be in

the form

`\ecvCEF{\level}{\desc}.` [§5.2]

`\ecvLeftColumnWidth{\width}`

Defined in: europecv class.

Sets the width of the left column. [§5.2]

`\ecvmothertongue[{\vspace}]{\language}`

Defined in: europecv class.

Starts the spoken language section and identifies your mother tongue. [§5.2]

`\ecvname{\name}`

Defined in: europecv class.

Specifies your name. [§5.2]

`\ecvnationality{\nationality}`

Defined in: europecv class.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
Specifies your nationality. [§5.2]	May only be used within the argument of <code>\ecvbeforepicture</code> or <code>\ecvafterpicture</code> to insert some vertical space. [§5.2]	A N
<code>\ecvpersonalinfo[<vspace>]</code>		B O
Defined in: europecv class.		C P
Typesets your personal details. [§5.2]		D Q
<code>\ecvpicture[<options>]{<image filename>}</code>	<code>\ecvtelephone[<mobile>]{<telephone>}</code>	E R
Defined in: europecv class.	Defined in: europecv class.	F S
Specifies the name of the file showing an image of yourself. [§5.2]	Specifies your telephone number and optionally your mobile phone number. [§5.2]	G T
<code>\ecvsection[<vspace>]{<title>}</code>	<code>\edef<(cs)>{<arg-syntax>}{<definition>}</code>	H U
Defined in: europecv class.	Defined in: T _E X primitive.	I V
Creates a section heading. [§5.2]	This locally defines the command <code><cs></code> to the full expansion of <code><definition></code> . Use <code>\xdef</code> for global definitions. [§2.1]	J W
<code>\ecvspace{<height>}</code>		K X
Defined in: europecv class.	<code>\EFC{<description>}{<foreign currency>}{<amount>}</code>	L Y
		M Z

Summary of Commands and Environments

`\{<conversion rate>\}\{<base currency result>\}`

Defined in: invoice package.

For use within the `invoice` environment, this command is used to specify a foreign expense. [§4.2]

`\EFCi\{<description>\}\{<foreign currency>\}\{<amount>\}`
`\{<conversion rate>\}\{<base currency result>\}`

Defined in: invoice package.

For use within the `invoice` environment, this command is like `\EFC` but although the amount is added to the total expense it's not itemized. [§4.2]

`\emailfrom\{<text>\}`

Defined in: newlfm class.

Specifies the sender's email address. [§3.3]

`\emph\{<text>\}`

Defined in: L^AT_EX Kernel.

Toggles the upright and italic/slanted rendering of `<text>`. (See **Volume 1** [93, §4.5.1].) [§2.1]

`\empty`

Defined in: L^AT_EX Kernel.

Does nothing. [§12.5]

`\encl\{<enclosures info>\}`

Defined in: Classes that define the `letter` environment.

Used to indicate any enclosures accompanying the letter. [§3.1]

`\enclist\{<text>\}`

Defined in: newlfm class.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
Specifies the list of enclosures. [§3.3]	Marks the end of the header code for the <code>longtable</code> environment. [§4.3]	A N
<code>\end{<env-name>}</code>	<code>\endinput</code>	B O
Defined in: L ^A T _E X Kernel.	Defined in: T _E X primitive.	C P
Ends an environment. (Must have a matching <code>\begin{env-name}</code> . See Volume 1 [93, §2.15].) [§1.2]	Stops reading the current file. Anything in the current file occurring after this command is skipped. [§7.3]	D Q
<code>\endfirsthead</code>	<code>\endlastfoot</code>	E R
Defined in: <code>longtable</code> package.	Defined in: <code>longtable</code> package.	F S
Marks the end of the header code for the first page of the <code>longtable</code> environment. [§4.3]	Marks the end of the footer code for the last page of the <code>longtable</code> environment. [§4.3]	G T
<code>\endfoot</code>	<code>\endtime{<time>}</code>	H U
Defined in: <code>longtable</code> package.	Defined in: <code>minutes</code> package.	I V
Marks the end of the footer code for the <code>longtable</code> environment. [§4.3]	Specifies the end time of the meeting. [§6.3]	J W
<code>\endhead</code>		K X
Defined in: <code>longtable</code> package.		L Y
		M Z

Summary of Commands and Environments

`\enspace`

Defined in: L^AT_EX Kernel.

Horizontal spacing command (half as wide as `\quad`). [§9.1]

`\begin{enumerate}`

Defined in: L^AT_EX Kernel.

Ordered list. Some packages, such as `paralist`, modify this environment to provide an optional argument that allows you to adjust the counter format. [§6.5]

`\preto{cs}{<code>}`

Defined in: etoolbox package.

Similar to `\preto` but expands `<code>`. Use `\xpreto` for a global assignment. [§2.1]

`\equal{<text1>}{<text2>}`

Defined in: ifthen package.

A test that can be used within the condition of `\ifthenelse` to determine if `<text1>` is the same as `<text2>`. (Both arguments are expanded.) [§2.7]

`\begin{europecv}`

Defined in: europecv class.

The body of the CV. [§5.2]

`\evensidemargin`

Defined in: L^AT_EX Kernel.

A length containing the horizontal offset for even pages. See `\hoffset`. [§11.0]

`\everypar{<code>}`

Defined in: T_EX primitive.

Indicates code to be performed at the start of every paragraph. [§6.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`exists(<ref>)`

Defined in: arara directive.

Evaluates to true if the given file doesn't exist. The argument `<ref>` may either be a string "`<extension>`" which indicates the file extension or a file reference `toFile("<filename>")`. [§1.2]

`\expandafter<token 1><token 2>`

Defined in: TeX primitive.

Equivalent to `<token 1>` expansion of `<token 2>`. [§2.7]

`\expandonce<cs>`

Defined in: etoolbox package.

Only permits one level of expansion of the command `<cs>`. [§2.1]

F

`\fbox{<text>}`

Defined in: L^AT_EX Kernel.

Puts a frame around its contents, prohibiting a line break in the contents. [§7.5]

`\Fee{<description>}{{<rate/unit>}}{<count>}`

Defined in: invoice package.

For use within the `invoice` environment, this command is used to specify a fee. [§4.2]

`\begin{figure}[<placement>]`

Defined in: Most classes except for those designed for correspondence or similarly restrictive documents.

Floats the contents to the nearest location according to the

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

preferred placement options, if possible. Within the environment, `\caption` may be used one or more times, as required. (See Volume 1 [93, §7.1].) [§2.1]

`\FilledSmallCircle`

Defined in: `ifsym` package with `geometry` option.

Produces a small filled circle.

[§11.1]

`\FirstLabel{\langle row \rangle}{\langle column \rangle}`

Defined in: `enlab` package.

Sets the starting label on a partially used sheet. [§3.6]

`\begin{flushright}`

Defined in: `LATEX` Kernel.

Aligns its contents flush-right and places a small vertical gap above and below the environment. [§10.3]

`\fontfamily{\langle name \rangle}`

Defined in: `LATEX` Kernel.

Sets the name of the current font family. (The change won't take effect until the next `\selectfont`.) For example, in text mode,

`\rmfamily` is equivalent to
`\fontfamily{cmr}\selectfont`
(unless it's been modified by a font package.) [§6.4]

`\fontseries{\langle weight \rangle}`

Defined in: `LATEX` Kernel.

Sets the name of the current font series. (The change won't take effect until the next `\selectfont`.) For example, in text mode,

`\bfseries` is equivalent to
`\fontseries{b}\selectfont`
(unless it's been modified by a font package.) [§6.4]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\footnote[⟨number⟩]{⟨text⟩}`

Defined in: L^AT_EX Kernel.

Inserts a footnote. [§2.1]

`\forcsvlist{⟨handler-cs⟩}{⟨item1, item2,...⟩}`

Defined in: etoolbox package.

This is like `\docslist` except that instead of using `\do` it uses `⟨handler-cs⟩`. [§2.7]

`\foreach{⟨variables⟩}[⟨options⟩] in {⟨list⟩}{⟨body⟩}`

Defined in: pgffor package.

Iterates through the `⟨list⟩` and assigns `⟨variables⟩` which can be used in `⟨body⟩`. The full syntax is quite complicated, so read the pgf manual for further details. The `\breakforeach` command can be used to prematurely terminate the

loop after the current iteration.

[§2.7]

`\foreachdataset{⟨cs⟩}{⟨body⟩}`

Defined in: probsoln package.

Iterates over all defined datasets and performs `⟨body⟩` at each iteration. [§9.3]

`\foreachproblem{⟨dataset⟩}{⟨body⟩}`

Defined in: probsoln package.

Iterates through the given dataset and does `⟨body⟩` at each iteration. Within `⟨body⟩`, `\thisproblem` may be used to display the current problem and `\thisproblemlabel` may be used to access the current problem label. [§9.3]

`\foreachsolution{⟨dataset⟩}`

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

{⟨body⟩}

Defined in: probsoln package.

Similar to `\foreachproblem` but only iterates through problems that contain the `onlysolution` environment. [§9.3]

`\forlistcsloop{⟨handler-cs⟩}{⟨list-csname⟩}`

Defined in: etoolbox package.

Similar to `\forlistloop` except the list control sequence name (without the leading backslash) is used. [§2.7]

`\forlistloop{⟨handler-cs⟩}{⟨list-cs⟩}`

Defined in: etoolbox package.

Similar to `\dolistloop` except it uses `⟨handler-cs⟩` instead of `\do` at each iteration. [§2.7]

`\begin{Form}[⟨parameters⟩]`

Defined in: hyperref package.

Environment containing interactive form elements. [§11.2]

`found(⟨ref⟩, ⟨expression⟩)`

Defined in: arara directive.

Evaluates to true if the given regular expression is found in the given file. The argument `⟨ref⟩` may either be a string "`⟨extension⟩`" which indicates the file extension or a file reference `toFile("⟨filename⟩")`. [§1.2]

`\FPrandom{⟨cs⟩}`

Defined in: fp package.

Generates a random number between 0 and 1 and assigns the result to the given control sequence `⟨cs⟩`. [§9.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\FPseed

Defined in: fp package.

A count register that stores the random generator seed. [§9.5]

\frac{\langle numerator \rangle}{\langle denominator \rangle}

Defined in: L^AT_EX Kernel (Math Mode).

Displays a fraction. [§9.3]

\begin{frame}

Defined in: beamer class.

Creates a slide (or possibly multiple slides if the frame contains overlays) [§8.0]

\frame{\langle text \rangle}

Defined in: L^AT_EX Kernel.

Puts a rectangular frame around *text*. Similar to \fbox but doesn't

insert a gap between *text* and the frame. The beamer class redefines this command for its frame environment. [§10.4]

\framebox[\langle width \rangle][\langle align \rangle]{\langle text \rangle}

Defined in: L^AT_EX Kernel.

Puts a frame around its contents, prohibiting a line break in the contents. [§10.1]

\framebox(\langle w \rangle,\langle h \rangle)[\langle align \rangle]{\langle text \rangle}

Defined in: picture environment.

Unlike the ordinary \framebox command, this version doesn't add any space between the frame and the text. [§10.1]

\framebreak

Defined in: flowfram package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Inserted at the point of the frame break when a paragraph spans two flow frames of unequal widths.
[§10.5]

`\framesubtitle{<subtitle>}`

Defined in: beamer class.

Subtitle for a frame. (For use within the `frame` environment.)
[§8.0]

`\frametitle{<title>}`

Defined in: beamer class.

Title for a frame. (For use within the `frame` environment. [§8.0])

G

`\ganttbar{<options>} {<text>} {<start-tss>} {<end-tss>}`

Defined in: pgfgantt package.
For use within the `ganttchart` environment, this draws a bar

representing a task or subtask.
[§12.4]

`\begin{ganttchart}[<options>]{<start>}{<end>}`

Defined in: pgfgantt.

Creates a Gantt chart. [§12.4]

`\ganttgroup{<options>} {<text>} {<start-tss>} {<end-tss>}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a group bar. [§12.4]

`\ganttlinkedbar{<options>} {<text>} {<start-tss>} {<end-tss>}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a linked bar representing a task or subtask.
[§12.4]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\ganttlinkedgroup[options]
{text}{{start-tss}}{{end-tss}}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a linked group bar. [§12.4]

`\ganttlinkedmilestone
[options]{{text}}{{tss}}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a linked milestone marker. [§12.4]

`\ganttmilestone[options]
{{text}}{{tss}}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a milestone marker. [§12.4]

`\ganttnewline[options]`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this starts a new row. [§12.4]

`\ganttset{options}`

Defined in: pgfgantt.

Sets the options governing the Gantt chart style. [§12.4]

`\ganttttitle[options]{{text}}
{{n}}`

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a single title element. [§12.4]

`\ganttttitlecalendar[options]
{{calendar-lines}}`

Defined in: pgfgantt package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

For use within the `ganttchart` environment, this draws a title calendar that spans the whole chart. [§12.4]

```
\ganttttitlecalendar*{<options>}  
  {<start-tss>} {<end-tss>} {<calendar  
  lines>}
```

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this draws a title calendar that spans the chart from `<start-tss>` to `<end-tss>`. [§12.4]

```
\ganttttitlelist [<options>]  
  {<list>} {<n>}
```

Defined in: pgfgantt package.

For use within the `ganttchart` environment, this iterates over `<list>` and draws a title element spanning `<n>` time slots. [§12.4]

```
\gappto{cs}{<code>}
```

Defined in: etoolbox package.

Global version of `\appto`. [§2.1]

```
\gdef{cs}{<arg-syntax>  
  {<definition>}}
```

Defined in: T_EX primitive.

As `\def` but the definition is global.
[§2.1]

```
\gDTLforeachbibentry  
  [<condition>] {<db-name>}  
  {<body>}
```

Defined in: databib package.

Global version of
`\DTLforeachbibentry`. The starred version is read-only. [§5.2]

```
\getflowbounds{<id>}
```

Defined in: flowfram package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Gets the location and size of the given flow frame. The results are stored in the lengths `\ffareaex`, `\ffareaey`, `\ffareawidth` and `\ffareaheight`. [§11.0]

`\getflowevenbounds{<id>}`

Defined in: `flowfram` package.

Gets the location and size of the given flow frame for even pages. The results are stored in the lengths `\ffareaex`, `\ffareaey`, `\ffareawidth` and `\ffareaheight`. [§11.0]

`\global<assignment>`

Defined in: `TeX` primitive.

An assignment prefix that indicates the following assignment shouldn't be confined to the current scope. [§2.1]

`\gpreto{<cs>}{<code>}`

Defined in: `etoolbox` package.

Global version of `\preto`. [§2.1]

`\gradetable[<orientation>]{<index-type>}`

Defined in: `exam` class.

Displays the grading table. [§9.1]

`\greetto{<text>}`

Defined in: `newlrm` class.

The salutation text. [§3.3]

`\guest{<names>}`

Defined in: `minutes` package.

Specifies the names of any guests present. [§6.3]

H

`\half`

Defined in: `exam` class.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
Indicates a half point [§9.1]	Similar to the <code>subitems</code> environment but is only displayed if the agenda class option is used. [§6.3]	A N
<code>\headline{\langle text \rangle}</code>		B O
Defined in: newlfm class.		C P
Specifies the subject of a press release. [§6.2]		D Q
<code>\hfill</code>		E R
Defined in: L ^A T _E X Kernel.		F S
Inserts a horizontal space that will expand to fit the available width. [§3.6]	This environment only displays its contents if the agenda class option is used. [§6.3]	G T
<code>\begin{hiddenitems}</code>		H U
Defined in: meetingmins class.		I V
Similar to the <code>items</code> environment but is only displayed if the agenda class option is used. [§6.3]	<code>\hideanswers</code>	J W
<code>\begin{hiddensubitems}</code>	Defined in: probsoln package.	K X
Defined in: meetingmins class.	Hides the solutions (from that point onwards). [§9.3]	L Y
		M Z
	<code>\hoffset</code>	
	Defined in: L ^A T _E X Kernel.	
	A length containing the horizontal offset. The left margin of a page is computed from <code>\hoffset</code> plus 1 in plus	

Summary of Commands and Environments

		Symbols
<code>\oddsidemargin</code> / <code>\evensidemargin</code> .	<code>\hypersetup{<key-val list>}</code>	A N
[§11.0]	Defined in: hyperref package.	B O
<code>\hrulefill</code>	Set up options. [§2.3]	C P
Defined in: L ^A T _E X Kernel.	I	D Q
Fills the remaining space with a line. [§9.1]	<code>if <condition></code>	E R
<code>\hspace{<length>}</code>	Defined in: arara directive.	F S
Defined in: L ^A T _E X Kernel.	Only run the application if <code><condition></code> is true. [§1.2]	G T
Inserts a horizontal gap of the given width. The unstarred version doesn't create a space if it occurs at the beginning or end of a paragraph. The starred version always creates a space. [§3.6]	<code>\ifbool{<name>}{{<true>}}{<false>}</code>	H U
<code>\Huge</code>	Defined in: etoolbox package.	I V
Defined in: Most document classes.	Expands to <code><true></code> if the boolean flag <code><name></code> is true, and to <code><false></code> otherwise. [§9.3]	J W
Switches to extra-huge sized text. [§10.3]	<code>\ifboolexpr{<expression>}{<true>}{<false>}</code>	K X
	Defined in: etoolbox package.	L Y
		M Z

Summary of Commands and Environments

Evaluates the `<expression>` and executes `<true>` if true, and `<false>` otherwise. The syntax for the expression is described in the etoolbox manual. [§2.9]

```
\ifcase<(number)> <(case0 code> \or  
<(case1 code> \or <(case2 code> \or  
... \else <(default code> \fi
```

Defined in: TeX primitive.

If `<(number)>` equals 0, performs `<(case0 code>`, if `<(number)>` equals 1, performs `<(case1 code>`, if `<(number)>` equals 2, performs `<(case2 code>`, etc. If none of the cases match, `<(default code>` is performed. The `\else <(default code>` part may be omitted. If `<(case 0 code>` starts with a number, insert `\relax` before it to prevent TeX from scanning it as part of `<(number)>`. [§7.3]

```
\ifcsdef{<(cs-name)>} {<true-part>} {<false-part>}
```

Defined in: etoolbox package.

Checks if the control sequence with the name `<(cs-name)>` exists. [§2.1]

```
\ifcsundef{<(cs-name)>} {<true-part>} {<false-part>}
```

Defined in: etoolbox package.

Checks if the control sequence whose name is given by `<(cs name)>` doesn't exist or is defined as `\relax`. [§2.1]

```
\ifdate{<(tests)>} {<(true-part)>}  
 {<(false-part)>}
```

Defined in: pgfcalendar package.

For use within the `<(code)>` part of `\pgfcalendar`. The same as

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\pgfcalendarifdate` for the current date. [§7.5]

`\ifdef{\cs}{\true-part}{\false-part}`

Defined in: etoolbox package.

Checks if the control sequence `\cs` exists. [§2.1]

`\ifdefempty{\cs}{\true}{\false}`

Defined in: etoolbox package.

If the control sequence `\cs` is empty this does `\true` otherwise it does `\false`. [§2.9]

`\ifdefstring{\cs}{\string}{\true}{\false}`

Defined in: etoolbox package.

If the control sequence `\cs` was defined to be `\string` this does

`\true` otherwise it does `\false`.
(No expansion is performed.) [§2.9]

`\ifinlist{\item}{\list-cs}{\true}{\false}`

Defined in: etoolbox package.

Checks if `\item` is included in one of etoolbox's internal list macros `\list-cs` and does `\true` if true, otherwise it does `\false`. No expansion is performed on `\item`. See also `\xifinlist` [§2.7]

`\ifinlistcs{\item}{\list-csname}{\true}{\false}`

Defined in: etoolbox package.

As `\ifinlist` but the control sequence name is supplied (without the backslash). [§2.7]

`\ifnum<num1> <comp> <num2>`

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\{true-part\}\else \{false-part\}\fi`

Defined in: TeX primitive.

Compares `\{num1\}` to `\{num2\}`. The comparison `\{comp\}` may be one of: `=` (equality), `<` (less than) or `>` (greater than). The `\else` `\{false-part\}` may be omitted. If `\{true-part\}` starts with a number, insert `\relax` before it to prevent TeX from scanning it as part of `\{num2\}`. [§2.7]

`\ifnumless{\{number1\}}{\{number2\}}{\{true-part\}}{\{false-part\}}`

Defined in: etoolbox package.

Checks if `\{number1\}` is less than `\{number2\}`. Both values should be integers. [§5.2]

`\ifodd{\{number\}} {\{odd code\}}\else`

`\{even code\}\fi`

Defined in: TeX primitive.

Tests if `\{number\}` is odd. If true this does `\{odd code\}` otherwise it does `\{even code\}`. The `\else` `\{even code\}` may be omitted. [§11.0]

`\ifshowanswers {\{true-part\}}{\{false-part\}}`

Defined in: probsoln package.

Tests if the solutions are displayed. [§9.3]

`\ifstrempty{\{string\}}{\{true-part\}}{\{false-part\}}`

Defined in: etoolbox package.

Tests if `\{string\}` is empty. (No expansion is performed on `\{string\}`.) [§7.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\ifstrequal{\langle string1 \rangle}{\langle string2 \rangle}`
`{\langle true-part \rangle}{\langle false-part \rangle}`

Defined in: etoolbox package.

Tests if `\langle string1 \rangle` is the same as `\langle string2 \rangle`. (No expansion is performed on `\langle string1 \rangle` or `\langle string2 \rangle`.) [§7.4]

`\ifthenelse{\langle condition \rangle}{\langle true \rangle}`
`{\langle false \rangle}`

Defined in: ifthen package.

If the condition is met, does `\langle true \rangle` otherwise does `\langle false \rangle`. The `\langle condition \rangle` must follow the syntax defined by the ifthen package. The datatool package provides additional commands that may be used in `\langle condition \rangle`. Note that in general, it's better to use the conditionals provided by the etoolbox package, but the optional

argument of `\DTLforeach` (and the starred version) requires the same format as the first argument of `\ifthenelse`. [§2.7]

`\ifthispageodd{\langle odd code \rangle}{\langle even code \rangle}`

Defined in: KOMA-Script classes.

Determines if the current page is odd or even. [§11.0]

`\ifundef{\langle cs \rangle}{\langle true-part \rangle}`
`{\langle false-part \rangle}`

Defined in: etoolbox package.

Checks if the control sequence `\langle cs \rangle` doesn't exist or is defined as `\relax`. [§2.1]

`\ignorespaces`

Defined in: T_EX primitive.

Used in begin environment code to suppress any spaces occurring at

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

the start of the environment (see also `\ignorespacesafterend`). [§9.3]

`\ignorespacesafterend`

Defined in: L^AT_EX Kernel.

Used in end environment code to suppress any spaces following the end of the environment. [§9.3]

`\iitem{<description>}{<amount>}`

Defined in: isodoc class.

Generates a row of data for use within `\itable`. [§4.1]

`\include{<filename>}`

Defined in: L^AT_EX Kernel.

Issues a `\clearpage`, creates an associated auxiliary file, inputs `<filename>` and issues another `\clearpage`. (See also `\input`.) [§13.0]

`\includegraphics[<option-list>]{<filename>}`

Defined in: graphicx package.

Inserts a graphics file into the document. Permitted file types depend on the output format. (PostScript (PS) and Encapsulated PostScript (EPS) for the DVI format. PDF, JPG and PNG for the PDF format (also EPS if the T_EX distribution permits on-the-fly `epstopdf` conversion). [§3.2]

`\includequestions[<options>]{<filenames>}`

Defined in: exsheets package.

Includes questions defined in the named files. [§9.2]

`\incorrectitem`

Defined in: probsoln package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

For use within the `\textenum`, this command may be used in place of `\item` to indicate an incorrect choice. If the solutions aren't displayed this command behaves the same as `\item`. [§9.3]

`\incorrectitemformat{\langle marker \rangle}`

Defined in: `probsoln` package.

The format used by

`\incorrectitem`. [§9.3]

`\begin{inparaenum}[\langle format \rangle]`

Defined in: `paralist` package.

An inline numbered list. (Similar to the `enumerate` environment, but the items don't start a new paragraph, unless you explicitly insert a paragraph break.) [§9.2]

`\input{\filename}`

Defined in: `LATEX Kernel`.

Reads in the contents of `\filename`. [§2.2]

`\InputIfFileExists{\langle file \rangle}{\langle true-part \rangle}{\langle false-part \rangle}`

Defined in: `LATEX Kernel`.

If the given file exists, this does `\true-part` and then loads the file. Otherwise it does `\false-part`. [§6.4]

`\inst{\langle text \rangle}`

Defined in: `beamer` class.

Used to prefix each institute listed in `\institute` when there are multiple institutes.

A corresponding `\inst{\langle number \rangle}` should be placed after the relevant author name within `\author`. [§8.0]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\institute{<text>}`

Defined in: Various classes or packages that have the concept of an institute in the title page.

Specifies the author's institution for use with `\maketitle` in a similar manner to `\title`. Some classes, such as beamer, also provide an optional argument for this command. [§8.0]

`\begin{invoice}{<base-currency>}{'<VAT>}'`

Defined in: invoice package.

The body of the invoice. [§4.2]

`\invoice[<options>]{<contents>}`

Defined in: isodoc class.

Creates an invoice. [§4.1]

`\itable{<contents>}`

Defined in: isodoc class.

Generates the table used within

`\invoice`. [§4.1]

`\item[<marker>]`

Defined in: L^AT_EX Kernel.

Specifies the start of an item in a list. (Only allowed inside one of the list making environments, such as `enumerate`.) [§5.1]

`\begin{itemize}`

Defined in: L^AT_EX Kernel.

Unordered list. [§8.1]

`\begin{items}`

Defined in: meetingmins class.

A numbered list for use within a section. Use the `subitems` environment for lists within subsections and sub-subsections. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\itotal[⟨tag⟩]{⟨amount⟩}`

Defined in: isodoc class.

Generates a row of for the total amount for use within `\itable`.

[§4.1]

`\itshape`

Defined in: L^AT_EX Kernel.

Switches to the italic form of the current font family, if it exists. See **Volume 1** [93, §4.5.1]. [§10.2]

J

`\jobname`

Defined in: T_EX primitive.

The current job name. This is usually the base name (without the .tex extension) of the main .tex file, but can be changed using T_EX's -jobname switch. [§5.2]

K

`\KOMAoption{⟨option⟩}{⟨value list⟩}`

Defined in: KOMA-Script classes.

Allows you to specify a list of values to the given multi-valued KOMA-Script option. [§3.2]

`\KOMAoptions{⟨option list⟩}`

Defined in: KOMA-Script classes.

Allows you to set one or more of the KOMA-Script options. [§3.2]

Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

L

`\label{⟨string⟩}`

Defined in: L^AT_EX Kernel.

Assigns a unique textual label linked to the most recently incremented cross-referencing counter in the current scope. (See **Volume 1** [93, §5.5].) [§1.2]

Summary of Commands and Environments

\labelenumi

Defined in: L^AT_EX Kernel.

Used by \item in the first level enumerate environment to display the label. [§6.5]

\labelenumii

Defined in: L^AT_EX Kernel.

Used by \item in the second level enumerate environment to display the label. [§6.5]

\labelenumiii

Defined in: L^AT_EX Kernel.

Used by \item in the third level enumerate environment to display the label. [§6.5]

\labelenumiv

Defined in: L^AT_EX Kernel.

Used by \item in the fourth level enumerate environment to display the label. [§6.5]

\LabelHeight

Defined in: envlab package.

A length register used to store the label height. [§3.6]

\labelsep

Defined in: L^AT_EX Kernel.

Length register used by list environments to store the size of the horizontal gap between the item marker and the following text. [§6.5]

\LabelWidth

Defined in: envlab package.

A length register used to store the label width. [§3.6]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\labelwidth

Defined in: L^AT_EX Kernel.

Length register used by list environments to store the label width. The value is typically set in the *<list declarations>* argument of the `list` environment. [§6.5]

\Large

Defined in: Most document classes.

Switches to extra-large sized text. [§2.1]

\begin{Large}

Defined in: Most document classes.

Sets its body in an extra-large sized font. [§2.1]

\large

Defined in: Most document classes.

Switches to large sized text. [§10.1]

\LayoutCheckField{\<label>} {<field>}

Defined in: hyperref package.

Lays out the label and associated check box. This just defaults to *<label> <field>*. [§11.2]

\LayoutChoiceField{\<label>} {<field>}

Defined in: hyperref package.

Lays out the label and associated choice field. This just defaults to *<label> <field>*. [§11.2]

\LayoutTextField{\<label>} {<field>}

Defined in: hyperref package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Lays out the label and associated text field. This just defaults to `\{label\} \{field\}`. [§11.2]

`\leftmargin`

Defined in: L^AT_EX Kernel.

Length register used by list environments to store the left margin width. The value is typically set in the `\{list declarations\}` argument of the `list` environment. [§6.5]

`\LenToUnit{\{length\}}`

Defined in: leaflet class.

May be used to specify a length instead of a value that's in terms of `\unitlength` for use within the `picture` environment. [§10.3]

`\let\{new cs\}\{org cs\}`

Defined in: T_EX primitive.

Gives `\{new cs\}` the same meaning as `\{org cs\}`. For example if I define:

`\newcommand{\mycmd}{stuff}` and then `\let\mynewcmd\mycmd` is like doing

`\newcommand{\mynewcmd}{stuff}`, but no existence check is performed. If I then redefine `\mycmd`, `\mynewcmd` still retains the old definition of `\mycmd`. Since there's no check to see if the new command is already defined, care is required to prevent accidentally overwriting a preexisting command. (The arguments `\{new cs\}` and `\{org cs\}` don't actually have to be control sequences. They can be any tokens, but that's beyond the scope of this book.) [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\letcs{\cs}{\cs-name}`

Defined in: etoolbox package.

Analogous to `\let` except that the name of the control sequence (without the initial backslash) is supplied for the second argument. [§2.1]

`\begin{letter}{\addressee}`

Defined in: Some classes used for writing letters, such as `scrltr2`.

Typesets its contents as correspondence. The KOMA-Script `scrltr2` class also allows an optional argument before the mandatory argument. [§3.1]

`\letter[\langle recipient-options \rangle]{\contents}`

Defined in: `isodoc` class.

Creates a letter. [§3.4]

`\lim`

Defined in: L^AT_EX Kernel (Math Mode).

Typesets lim function name (may have limits via `_` or `^`). [§9.3]

`\line(\langle h \rangle,\langle v \rangle){\langle length \rangle}`

Defined in: L^AT_EX Kernel.

For use within the argument of `\put`, this draws a straight line of the given length whose horizontal and vertical extent (gradient vector) is given by $(\langle h \rangle,\langle v \rangle)$. [§10.1]

`\lineskip`

Defined in: L^AT_EX Kernel.

A T_EX primitive that stores the interline glue. [§3.6]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\linewidth

Defined in: L^AT_EX Kernel.

A length containing the desired current line width. This is usually the width of the typeblock, but inside a `minipage` or `\parbox` it will be the width the box. Note that the actual contents of the line may fall short of the line width (underfull `hbox`) or extend beyond it (overfull `hbox`). [§4.3]

\lipsum[⟨selection⟩]

Defined in: lipsum package.

Produces dummy text for testing purposes. The optional argument specifies which paragraphs to display. (There are 250 predefined paragraphs). By default this command displays the first seven paragraphs, but this may be

changed via the optional argument, which may be either a single number or a range. The starred version suppresses the paragraph breaks. [§6.4]

\begin{list}{⟨label⟩}{⟨list declarations⟩}

Defined in: L^AT_EX Kernel.

A generic list environment [§6.5]

\listadd{⟨list-cs⟩}{⟨item⟩}

Defined in: etoolbox package.

Appends the given ⟨item⟩ to the list control sequence ⟨list-cs⟩. A blank item is not added. No expansion is performed on the item. [§2.7]

\listcsadd{⟨list-csname⟩}{⟨item⟩}

Defined in: etoolbox package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Similar to `\listadd` except the name (without the preceding backslash) of the list control sequence is used. [§2.7]

`\listcseadd{\langle list-csname \rangle}{\langle item \rangle}`

Defined in: etoolbox package.

Similar to `\listadd` except the name (without the preceding backslash) of the list control sequence is used. [§2.7]

`\listcsgadd{\langle list-csname \rangle}{\langle item \rangle}`

Defined in: etoolbox package.

As `\listcsadd` but the assignment is global. [§2.7]

`\listcsxadd{\langle list-csname \rangle}{\langle item \rangle}`

Defined in: etoolbox package.

As `\listcseadd` but the assignment is global. [§2.7]

`\listead{\langle list-cs \rangle}{\langle item \rangle}`

Defined in: etoolbox package.

Appends the expansion of `\langle item \rangle` to the list control sequence `\langle list-cs \rangle`. A blank item is not added. [§2.7]

`\listgadd{\langle list-cs \rangle}{\langle item \rangle}`

Defined in: etoolbox package.

As `\listadd` but the assignment is global. [§2.7]

`\listofchanges[{\langle options \rangle}]`

Defined in: changes package.

Print a list or summary of the changes. At least two L^AT_EX runs are required to make the list appear. [§13.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\listofdecisions

Defined in: minutes package.

Displays a list of decisions. [§6.3]

\listoffigures

Defined in: Most classes that have the concept of document structure.

Inserts the list of figures. A second (possibly third) run is required to ensure the page numbering is correct. [§13.1]

\listoftables

Defined in: Most classes that have the concept of document structure.

Inserts the list of tables. A second (possibly third) run is required to ensure the page numbering is correct. [§6.3]

\listxadd{\langle list-cs \rangle}{\langle item \rangle}

Defined in: etoolbox package.

As \listadd but the assignment is global. [§2.7]

\llap{\langle text \rangle}

Defined in: L^AT_EX Kernel.

Places \langle text \rangle to the left of the reference point without taking up any space. [§11.1]

\loadallproblems[\langle dataset \rangle]{\langle filename \rangle}

Defined in: probsoln package.

Loads all problems defined in \langle filename \rangle and appends them to the specified data set in the order in which they are defined in the file. [§9.3]

\LoadClass[\langle options \rangle]{\langle name \rangle}[\langle version \rangle]

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Used in class files to load another class. [§11.1]

\loadexceptproblems[*dataset*]{*n*}{*filename*}

Defined in: probsoln package.

Loads the problems defined in the given filename except those listed in *exception list* and adds them to the given dataset. [§9.3]

\loadrandomexcept[*dataset*]{*n*}{*filename*}{*exception-list*}

Defined in: probsoln package.

Loads *n* randomly selected problems defined in the given filename excluding those listed in *exception list* and adds them to the given dataset. [§9.3]

\loadrandomproblems[*dataset*]{*n*}{*filename*}

Defined in: probsoln package.

Loads *n* randomly selected problems defined in the given filename and adds them to the given dataset. [§9.3]

\loadselectedproblems[*dataset*]{*labels*}{*filename*}

Defined in: probsoln package.

Loads the listed problems defined in the given filename and adds them to the given dataset. [§9.3]

\location{*text*}

Defined in: letter class.

Specifies the sender's additional location information. [§3.1]

\location{*place*}

Defined in: minutes package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Specifies the location of the meeting. [§6.3]

`\long<assignment>`

Defined in: TeX primitive.

An assignment prefix that indicates the following assignment (such as `\def`) should define a long command. [§2.1]

`\begin{longtable}{<horizontal alignment>} {<column specifiers>}`

Defined in: longtable package.

Like a combination of the `table` and `tabular` environments, except it can span multiple pages. [§2.6]

`\loop<code>\if... \repeat`

Defined in: L^AT_EX Kernel.

Repeats code while the given condition is true. [§2.7]

`\lstinputlisting{<options>} {<filename>}`

Defined in: listings package.

Reads in `<filename>` and typesets the contents as displayed code. [§9.4]

`\begin{lstlisting}{<options>}`

Defined in: listings package.

Typesets the contents of the environment as displayed code. [§8.0]

M

`\makeatletter`

Defined in: L^AT_EX Kernel.

Changes the `category code` of the at character (@) to “letter” so that it can be used in control sequence names. See also `\makeatother`. [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\makeatother`

Defined in: L^AT_EX Kernel.

Changes the [category code](#) of the at character (@) to “other”. This means that it can no longer be used in control sequence names.

See also `\makeatletter`. [§2.1]

`\makebox[⟨width⟩][⟨align⟩]{⟨text⟩}`

Defined in: L^AT_EX Kernel.

Puts $\langle text \rangle$ in a box, prohibiting a line break in the contents. [§9.1]

`\makebox[⟨w⟩,⟨h⟩][⟨align⟩]{⟨text⟩}`

Defined in: [picture](#) environment.

Creates a box picture object without a frame. [§10.1]

`\MakeButtonField{⟨text⟩}`

Defined in: hyperref package.

The format for push button labels.

[§11.2]

`\MakeCheckField{⟨width⟩}{⟨height⟩}`

Defined in: hyperref package.

The display for check fields.
[§11.2]

`\MakeChoiceField{⟨width⟩}{⟨height⟩}`

Defined in: hyperref package.

The display for choice fields.
[§11.2]

`\makelabels`

Defined in: letter class and envlab package (preamble only).

Switches on the label-generating function. [§3.6]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\MakeRadioField{<width>}{<height>}`

Defined in: hyperref package.

The display for radio fields. [§11.2]

`\MakeTextField{<width>}{<height>}`

Defined in: hyperref package.

The display for text fields. [§11.2]

`\maketitle`

Defined in: Most classes that have the concept of a title page.

Generates the title page (or title block). This command is usually placed at the beginning of the document environment. [§5.1]

`\MakeUppercase{<text>}`

Defined in: L^AT_EX Kernel.

Converts its argument to upper case. [§3.6]

`\marginpar[<left>]{<text>}`

Defined in: L^AT_EX Kernel.

Puts $\langle text \rangle$ in the margin. If the document provides left and right margins (for example, a two-sided document) $\langle left \rangle$ indicates the text to use if the margin is on the left and $\langle text \rangle$ indicates the text to use if the margin is on the right. [§6.5]

`\mbox{<text>}`

Defined in: L^AT_EX Kernel.

Ensures that the given text doesn't contain a line break. [§7.5]

`\medskip`

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Inserts a medium-sized vertical space. The size is given by the length `\medskipamount`. [§12.3]

`\midrule[<wd>]`

Defined in: `booktabs` package.

Horizontal rule to go below headings row of a `tabular` environment. [§2.6]

`\begin{minipage}[<pos>][<height>][<width>]`

Defined in: `LATEX` Kernel.

Makes a box with line-wrapped contents. (See also `\parbox`.) [§3.6]

`\begin{Minutes}{<title>}`

Defined in: `minutes` package.

Contains the minutes from a meeting. [§6.3]

`\minutesdate{<date>}`

Defined in: `minutes` package.

Specifies the date of the meeting. [§6.3]

`\minutetaker{<name>}`

Defined in: `minutes` package.

Specifies the name of the minute taker. [§6.3]

`missing(<ref>)`

Defined in: `arara` directive.

Evaluates to true if the given file doesn't exist. The argument `<ref>` may either be a string "`<extension>`" which indicates the file extension or a file reference `toFile("<filename>")`. [§1.2]

`\missing[<excused-names>]{<no-excuse>}`

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`names}`

Defined in: minutes package.

List of absentees. [§6.3]

`\missingExcused{<names>}`

Defined in: minutes package.

List of absentees who gave an excuse. [§6.3]

`\missingNoExcuse{<names>}`

Defined in: minutes package.

List of absentees who didn't give an excuse. [§6.3]

`\mlabel{<from-address>}{<to-address>}`

Defined in: envlab package.

Manually creates an address label. [§3.6]

`\moderation{<name>}`

Defined in: minutes package.

Specifies the name of the meeting moderator. [§6.3]

`\month`

Defined in: T_EX primitive.

The current month number. [§7.2]

`\multicolumn{<cols-spanned>}{<col-specifier>}{<text>}`

Defined in: L^AT_EX Kernel.

Spans multiple columns in a tabular-style environment. See Volume 1 [93, §4.6.2]. [§2.7]

`\multiply<register> by <value>`

Defined in: T_EX primitive.

Multiplies the value stored in `<register>` by `<value>`. The `by` keyword may be omitted. [§2.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\multiput(<x>,<y>)(<inc-x>,<inc-y>){<n>}{<object>}`

Defined in: L^AT_EX Kernel.

For use in the `picture` environment, this puts $\langle n \rangle$ copies of $\langle object \rangle$, starting at position $(\langle x \rangle, \langle y \rangle)$ and advancing the position by $(\langle inc-x \rangle, \langle inc-y \rangle)$ each time. [§10.1]

N

`\name{<text>}`

Defined in: letter class.

Specifies the sender's name. [§3.1]

`\namefrom{<name>}`

Defined in: newlfm class.

Specifies the sender's name. [§3.3]

`\nameto{<name>}`

Defined in: newlfm class.

Specifies the recipient's name.
[§3.3]

`\NeedsTeXFormat{<format>}
[<version>]`

Defined in: L^AT_EX Kernel.

This should be the first statement of any class or package identifying the required T_EX format. For a L^AT_EX₂ ϵ class or package this should be LaTeX2e. (Other formats may not define this command.)
[§7.3]

`\newbool{<name>}`

Defined in: etoolbox package.

Defines a new boolean flag called $\langle name \rangle$. [§9.4]

`\newboolean{<name>}`

Defined in: ifthen package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Defines a new boolean variable.

[§9.4]

`\newcommand{\langle cmd \rangle}{\langle n-args \rangle}{\langle default \rangle}{\langle text \rangle}`

Defined in: L^AT_EX Kernel.

Defines a new command. (See
Volume 1 [93, §8].) [§2.1]

`\newcount\langle cs \rangle`

Defined in: T_EX primitive.

Defines a new count register.
(Note to be confused with the L^AT_EX
`\newcounter` command.) [§2.1]

`\newcounter{\langle counter \rangle}{\langle outer-counter \rangle}`

Defined in: L^AT_EX Kernel.

Defines a new counter (see
Volume 1 [93, §11]). [§2.1]

`\newdynamicframe[\langle page-list \rangle]{\langle width \rangle}{\langle height \rangle}{\langle x \rangle}{\langle y \rangle}{\langle label \rangle}`

Defined in: flowfram package.

Defines a new dynamic frame.
The starred version adds a
rectangular border to the frame.
[§10.5]

`\newenvironment{\langle env-name \rangle}{\langle n-args \rangle}{\langle default \rangle}{\langle begin-code \rangle}{\langle end-code \rangle}`

Defined in: L^AT_EX Kernel.

Defines a new environment. [§11.1]

`\newflowframe[\langle page-list \rangle]{\langle width \rangle}{\langle height \rangle}{\langle x \rangle}{\langle y \rangle}{\langle label \rangle}`

Defined in: flowfram package.

Defines a new flow frame. The
starred version adds a rectangular
border to the frame. [§10.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\newglossaryentry{\langle label \rangle}{\langle key-val list \rangle}`

Defined in: glossaries package.

Defines a new glossary entry or term. [§2.1]

`\newlength{\length cs}`

Defined in: L^AT_EX Kernel.

Defines a new length register called `\length cs`. [§6.5]

`\begin{newlfm}`

Defined in: newlfm class.

The body of the letter, fax or memo. [§3.3]

`\newlfmP{\langle option list \rangle}`

Defined in: newlfm class.

Sets the given options. [§3.3]

`\newline`

Defined in: L^AT_EX Kernel.

Forces a line break. [§5.2]

`\newpage`

Defined in: L^AT_EX Kernel.

Forces a page break leaving a ragged bottom. [§10.3]

`\newproblem[{\langle n \rangle}][{\langle default-args \rangle}]{\langle label \rangle}{\langle problem \rangle}{\langle solution \rangle}`

Defined in: probsoln package.

A shortcut that defines a problem with an associated solution using the `defproblem`, `onlysolution` and `solution` environments. [§9.3]

`\newproblem*[{\langle n \rangle}][{\langle default-args \rangle}]{\langle label \rangle}{\langle problem \rangle}`

Defined in: probsoln package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

A shortcut that defines a problem using the `defproblem` environment. [§9.3]

```
\newstaticframe[<page-list>]
{<width>}{<height>}{<x>}{<y>}
[<label>]
```

Defined in: `flowfram` package.

Defines a new static frame. The starred version adds a rectangular border to the frame. [§10.5]

```
\newwatermark[<options>]
{<mark>}
```

Defined in: `xwatermark` package.

Specifies a watermark. [§6.4]

```
\nextmeeting{<date and time>}
```

Defined in: `meetingmins` class.

Displays the next meeting date and time. [§6.3]

`\noaddpoints`

Defined in: `exam` class.

Disable the point-totalling commands. [§9.1]

```
\NoBgThisPage
```

Defined in: `background` package.

Indicates the background shouldn't be displayed on the current page. (For use with the `all pages` option.) [§6.4]

```
\nobibliography{<bib-list>}
```

Defined in: `bibentry` package.

Analogous to `\bibliography`, this command writes the information to the `.aux` file that's required by BibTeX but doesn't display anything in the document.

Individual bibliography entries can

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

then be displayed using `\bibentry`.
[§5.2]

`\nocite{<key list>}`

Defined in: L^AT_EX Kernel.

Like `\cite` except that it doesn't produce any text. The `<key list>` may be just an asterisk * to indicated that all citations in the .bib file (or files) should be included in the bibliography. [§5.1]

`\node[<options>] (<label>){<text>};`

Defined in: tikz package.

Node specification, for use within the `tikzpicture` environment. [§7.5]

`\nodepart{<part>}`

Defined in: tikz package.

For use within the contents of a split node, this moves from the

current split to the split identified by `<part>`. [§7.5]

`\noexpand{<token>}`

Defined in: T_EX primitive.

Prevents `<token>` from being expanded in an expandable context. [§2.1]

`\noindent`

Defined in: L^AT_EX Kernel.

Suppress the indentation that would usually occur at the start of the next paragraph. [§9.1]

`\noprintanswers`

Defined in: exam class.

Disable (don't show) the solutions.
[§9.1]

`\normalsize`

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Switches to normal sized text.

[§10.3]

`\number<num>`

Defined in: T_EX primitive.

Displays the decimal value of `<num>`. (Any redundant leading zeros are stripped.) [§2.1]

`\numexpr<integer expression>`

Defined in: ε-T_EX primitive.

Expands to the value given by the integer expression. [§2.1]

`\numparts`

Defined in: exam class.

The number of parts. [§9.1]

`\numpoints`

Defined in: exam class.

The total number of points. [§9.1]

`\numquestions`

Defined in: exam class.

The number of questions. [§9.1]

`\numsubparts`

Defined in: exam class.

The number of sub-parts. [§9.1]

`\numsubsubparts`

Defined in: exam class.

The number of sub-sub-parts.
[§9.1]

Symbols

A N

B O

C P

D Q

E R

F S

G T

H U

I V

J W

K X

L Y

M Z

O

`\oddsidemargin`

Defined in: L_AT_EX Kernel.

A length containing the horizontal offset for odd pages. See

`\hoffset`. [§11.0]

Summary of Commands and Environments

\onecolumn

Defined in: L^AT_EX Kernel.

Issues a page break and switches to one column mode. [§10.5]

\begin{oneparcheckboxes}

Defined in: exam class.

Checkbox choices are listed inline.
Items are specified via \choice.
[§9.1]

\begin{oneparchoices}

Defined in: exam class.

Labelled choices are listed inline.
Items are specified via \choice.
[§9.1]

\begin{onlyproblem}[\langle option\rangle]

Defined in: probsoln package.

Only displays its contents if the showanswers boolean flag is off.
[§9.3]

\begin{onlysolution}[\langle option\rangle]

Defined in: probsoln package.

Only displays its contents if the showanswers boolean flag is on.
[§9.3]

\opening{\langle salutation\rangle}

Defined in: Classes that define the letter environment.

Typesets the salutation at the start of the letter. [§3.1]

\opinion{\langle main\rangle}{\langle differing\rangle}

Defined in: minutes package.

Indicates a discussion on an opinion. [§6.3]

\begin{Opinions}

Defined in: minutes package.

A list like environment to format opinions. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\oval(<w>,<h>) [<segment>]`

Defined in: L^AT_EX Kernel.

For use in the argument of `\put`, this command draws an oval of the given width and height. The optional argument indicates to only draw a quarter or half oval. [§10.1]

P

`\p@enumi`

Defined in: L^AT_EX Kernel.

Prefixed used when cross-referencing the enumi counter. [§6.5]

`\p@enumii`

Defined in: L^AT_EX Kernel.

Prefixed used when cross-referencing the enumii counter. [§6.5]

`\p@enumiii`

Defined in: L^AT_EX Kernel.

Prefixed used when cross-referencing the enumiii counter. [§6.5]

`\p@enumiv`

Defined in: L^AT_EX Kernel.

Prefixed used when cross-referencing the enumiv counter. [§6.5]

`\pagestyle{<style>}`

Defined in: L^AT_EX Kernel.

Sets the style of the headers and footers. [§10.1]

`\paperheight`

Defined in: L^AT_EX Kernel.

A length containing the total height of the page. See also `\textheight`. [§10.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\paperwidth

Defined in: L^AT_EX Kernel.

A length containing the total width of the page. See also \textwidth.
[§10.3]

\par

Defined in: T_EX primitive (context-dependent).

Insert a paragraph break. Usually just a blank line in the source code is used instead of \par. This command is typically only used within command or environment definitions or when it's important to remind authors that a paragraph break is required at a certain point where a blank line may appear accidental or may be overlooked. (For example, in {\centering Some centred

text.\par} it's a reminder that there must be a paragraph break before the closing } to ensure the \centering declaration has an effect.) [§1.0]

\paragraph[<short-title>]{<title>}

Defined in: Most classes that have the concept of document structure.

Inserts a subsubsubsection header. Most classes default to an unnumbered running header for this sectional unit. [§6.5]

\parbox[<pos>][<height>]{<width>}{<text>}

Defined in: L^AT_EX Kernel.

Makes a box with line-wrapped contents. (More restrictive than minipage.) [§7.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\parindent

Defined in: L^AT_EX Kernel.

A length register that stores the indentation at the start of paragraphs. [§2.1]

\parshape=⟨n⟩ ⟨i₁⟩ ⟨l₁⟩ … ⟨i_n⟩ ⟨l_n⟩

Defined in: T_EX primitive.

Creates a shaped paragraph.

[§10.5]

\part{⟨points⟩}

Defined in: exam class.

Inside the `parts` environment, this command starts a new numbered question part, with optionally the number of points the part is worth. Outside the `parts` environment this command works as a standard sectioning command. [§9.1]

\part[⟨short-title⟩]{⟨title⟩}

Defined in: Most classes that have the concept of document structure.

Inserts a part sectional unit. [§9.1]

\participant{⟨names⟩}

Defined in: minutes package.

Specifies the names of the people present. [§6.3]

\begin{parts}

Defined in: exam class.

Contains all the question parts.
[§9.1]

\PassOptionsToClass{⟨option-list⟩}{⟨class⟩}

Defined in: L^AT_EX Kernel.

Passes the given options to the given class. (Must be used before the class is loaded.) [§11.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\path <specification>;`

Defined in: tikz package.

For use within the `tikzpicture` environment. [§7.5]

`\pdfcreationdate`

Defined in: PDFTeX primitive.

The date and time at the start of the current TeX run. This expands to `D:<YYYY><MM><DD><hh><mm><ss><time zone>`. This primitive is also available with LuaTeX but not XeTeX. [§7.4]

`\pdffilemoddate{<filename>}`

Defined in: PDFTeX primitive.

The modification date and time of the file `<filename>`. This expands to `D:<YYYY><MM><DD><hh><mm><ss><time zone>`. Unlike

`\pdfcreationdate`, this primitive isn't provided by LuaTeX. [§7.4]

`\pgfcalendar{<prefix>}{<start-date>}{<end-date>}{<code>}`

Defined in: pgfcalendar package.

A for loop that iterates from `<start-date>` to `<end-date>` and performs `<code>` at each iteration. [§7.5]

`\pgfcalendarbeginiso`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. The `<start-date>` parameter in ISO format. [§7.5]

`\pgfcalendarbeginjulian`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. The `<start-date>`

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

parameter converted to a Julian day number. [§7.5]

`\pgfcalendarcurrentday`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. This expands to the day of the month for the current iteration. [§7.5]

`\pgfcalendarcurrentjulian`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. This is a T_EX count register that holds the Julian day number for the current iteration. [§7.5]

`\pgfcalendarcurrentmonth`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. This expands to the

month for the current iteration. [§7.5]

`\pgfcalendarcurrentweekday`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. This expands to the week day number for the current iteration (0 for Monday, 1 for Tuesday, etc). [§7.5]

`\pgfcalendarcurrentyear`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. This expands to the year for the current iteration. [§7.5]

`\pgfcalendardatetojulian`
`{<date>} {<register>}`

Defined in: pgfcalendar package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Converts the specified date into a Julian day number and stores the result in the given register. [§7.2]

`\pgfcalendarendiso`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. The `<end-date>` parameter in ISO format. [§7.5]

`\pgfcalendarendjulian`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. The `<end-date>` parameter converted to a Julian day number. [§7.5]

`\pgfcalendarifdate{<date>}{{<test>}}{<true-part>}{<false-part>}`

Defined in: pgfcalendar package.

Tests the specified date and does `<true-part>` if the test succeeds otherwise it does `<false-part>`. [§7.2]

`\pgfcalendarjuliantodate`
`{<Julian-day>}{{<year-cs>}}`
`{<month-cs>}{{<day-cs>}}`

Defined in: pgfcalendar package.

Converts a Julian day number into an ISO-date. The resulting numbers are stored in the `<year-cs>`, `<month-cs>` and `<day-cs>` control sequences. [§7.2]

`\pgfcalendarjuliantoweekday`
`{<Julian-day>}{{<register>}}`

Defined in: pgfcalendar package.

Converts a Julian day number into a week day number (0 for Monday, 1 for Tuesday, etc). The

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

resulting number is stored in the given register. [§7.2]

`\pgfcalendarmonthname{<month-number>}`

Defined in: pgfcalendar package.
Expands to a textual representation of the month. [§7.3]

`\pgfcalendarmonthshortname{<month-number>}`

Defined in: pgfcalendar package.
Expands to an abbreviated textual representation of the month. [§7.3]

`\pgfcalendarprefix`

Defined in: pgfcalendar package.
For use within the `<code>` part of `\pgfcalendar`. The `<prefix>` parameter. [§7.5]

`\pgfcalendarshorthand{<kind>} {<representation>}`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. Expands to a representation of the current day, month, year or day of week. [§7.5]

`\pgfcalendarsuggestedname`

Defined in: pgfcalendar package.

For use within the `<code>` part of `\pgfcalendar`. If the `<prefix>` parameter is empty, this command expands to empty otherwise it expands to `<prefix>-<YYYY>-<MM>-<DD>`, where `<YYYY>` is the current year, `<MM>` is the current month and `<DD>` is the current day of the month. [§7.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\pgfcalendarweekdayname{<week day number>}`

Defined in: pgfcalendar package.

Expands to a textual representation of the day of week. [§7.3]

`\pgfcalendarweekdayshortname{<week day number>}`

Defined in: pgfcalendar package.

Expands to an abbreviated textual representation of the day of week. [§7.3]

`\pgfkeys{<options>}`

Defined in: pgfkeys package.

Sets the options related to the pgf package and any related packages that use pgf's option interface.

[§12.5]

`\pgfmathdeclarerandomlist{<list name>}{{<item1>}{<item2>}...}`

Defined in: pgfmath package.

Defines a list that can have items randomly selected from it (using `\pgfmathrandomitem`). [§9.5]

`\pgfmathparse{<expression>}`

Defined in: pgfmath package.

Parses the given mathematical expression and stores the result in `\pgfmathresult`. [§9.5]

`\pgfmathprintnumber{<number>}`

Defined in: pgfmath package.

Pretty-prints the given number. [§12.5]

`\pgfmathrandominteger{<cs>}{{<minimum>}}{<maximum>}`

Defined in: pgfmath package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Defines the control sequence `\cs` to be a pseudo-randomly generated number between `\minimum` and `\maximum` (inclusive). [§9.5]

`\pgfmathrandomitem{\cs}{\listname}`

Defined in: pgfmath package.

Randomly select an item from the given list, which should have previously been declared using `\pgfmathdeclarerandomlist`. [§9.5]

`\pgfmathsetseed{n}`

Defined in: pgfmath package.

Sets the seed for the random number generator. [§9.5]

`\pgfplotsset{options}`

Defined in: pgfplots package.

Sets the options governing plot styles. [§12.5]

`\phonefrom{<text>}`

Defined in: newlrm class.

Specifies the sender's phone number. [§3.3]

`\begin{picture}(<width>,<height>)(<llx>,<lliy>)`

Defined in: L^AT_EX Kernel.

An environment that produces a box with the given dimensions (specified in terms of `\unitlength`). The second argument `\llx,\lliy` is optional and specifies the co-ordinates of the lower left corner, (0,0) if omitted.

The contents of the environment should consist of commands such as `\put` that puts text or lines in the box. [§10.0]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\pie[options]{list}`

Defined in: pgf-pie package.

For use within the `tikzpicture` environment. Draws a pie chart.
[§12.2]

`\points{n}`

Defined in: exsheets package.

Displays the number of points.
The starred version omits the unit.
[§9.2]

`\pointsdroppedatright`

Defined in: exam class.

Switches off the automatic point placement. (Used with
`\droppoints`.) [§9.1]

`\pointsinmargin`

Defined in: exam class.

Puts the points in the left margin.

[§9.1]

`\pointsinrightmargin`

Defined in: exam class.

Puts the points in the left margin.

[§9.1]

`\pointsum`

Defined in: exsheets package.

Displays the total number of points, excluding bonus points.
(Requires two L^AT_EX runs to ensure it's up to date.) The starred version omits the unit. [§9.2]

`\begin{Postscript}`

Defined in: minutes package.

Displays additional information that doesn't form part of the minutes. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

		Symbols
<code>\postscript{\langle text\rangle}</code>	<code>\PRcompany{\langle name\rangle}</code>	A N
Defined in: minutes package.	Defined in: pressrelease class.	B O
Displays additional information that doesn't form part of the minutes. [§6.3]	Specifies the company's name. [§6.2]	C P
<code>\pounds</code>	<code>\PRdepartment{\langle name\rangle}</code>	D Q
Defined in: L ^A T _E X Kernel.	Defined in: pressrelease class.	E R
Pound £ symbol. This robust command may be used in math or text mode. [§2.2]	Specifies the department's name. [§6.2]	F S
<code>\ppsitem{\langle text\rangle}</code>	<code>\PRemail{\langle address\rangle}</code>	G T
Defined in: newlfm class.	Defined in: pressrelease class.	H U
Specifies the PPS line. [§3.3]	Specifies the company's email address. [§6.2]	I V
<code>\PRaddress{\langle address\rangle}</code>	<code>\begin{pressrelease}</code>	J W
Defined in: pressrelease class.	Defined in: pressrelease class.	K X
Specifies the company's address. [§6.2]	Creates a press release. [§6.2]	L Y
	<code>\preto{\cs}{\langle code\rangle}</code>	M Z
	Defined in: etoolbox package.	

Summary of Commands and Environments

Prepends `<code>` to the definition of the control sequence `<cs>`. Use `\gpreto` for a global assignment. [§2.1]

`\PRfax{<number>}`

Defined in: pressrelease class.

Specifies the company's fax number. [§6.2]

`\PRheadline{<text>}`

Defined in: pressrelease class.

Specifies the headline. [§6.2]

`\PRhours{<times>}`

Defined in: pressrelease class.

Specifies the company's opening hours. [§6.2]

`\PrintAddress{<address>}`

Defined in: envlab package.

Used internally to typeset the recipient's address. [§3.6]

`\printanswers`

Defined in: exam class.

Enable (show) the solutions. [§9.1]

`\PrintBigLabel{<from-address>} {<to-address>}`

Defined in: envlab package.

Used internally to typeset the big labels. [§3.6]

`\PrintReturnAddress{<address>}`

Defined in: envlab package.

Used internally to typeset the sender's address. [§3.6]

`\printsolutions[<settings>]`

Defined in: exsheets package.

Prints all the solutions. [§9.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\priormins

Defined in: meetingmins class.

Displays the text “The minutes of the previous meeting were approved”. [§6.3]

\PRlocation{<location>}

Defined in: pressrelease class.

Specifies the company's location.

[§6.2]

\PRlogo{<code>}

Defined in: pressrelease class.

Specifies the code to use to produce the logo. Typically, this will usually just involve the

\includegraphics command.

[§6.2]

\PRmobile{<number>}

Defined in: pressrelease class.

Specifies the company's mobile phone number. [§6.2]

\Pro

Defined in: minutes package.

For use within the Argumentation environment, this indicates the start of an important item in favour of the argument. [§6.3]

\pro

Defined in: minutes package.

For use within the Argumentation environment, this indicates the start of an item in favour of the argument. [§6.3]

\ProbSolnFragileExt

Defined in: probsoln package.

Stores the extension of the temporary file used by probsoln to

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

work with verbatim code.
(Defaults to `verb`.) [§9.3]

`\ProbSolnFragileFile`

Defined in: `probsoln` package.

Stores the basename of the temporary file used by `probsoln` to work with verbatim code.
(Defaults to `\jobname`.) [§9.3]

`\ProcessOptions`

Defined in: L^AT_EX Kernel.

Process all the declare options.
[§11.1]

`\ProjectTitle{\langle title \rangle}`

Defined in: `invoice` package.

For use within the `invoice` environment, this command is used to specify the project title.
[§4.2]

`\begin{proof}[\langle proof name \rangle]`

Defined in: `beamer` class.

An environment for typesetting proofs. [§8.0]

`\protect<command>`

Defined in: L^AT_EX Kernel.

Used in a moving argument to prevent a fragile command from expanding. [§9.4]

`\protected@csedef{\langle cs-name \rangle}{\langle arg-syntax \rangle}{\langle definition \rangle}`

Defined in: `etoolbox` package.

Similar to `\protected@edef` but the name of the control sequence (without the leading backslash) is used. [§2.1]

`\protected@csxdef{\langle cs-name \rangle}{\langle arg-syntax \rangle}{\langle definition \rangle}`

Defined in: `etoolbox` package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Similar to `\protected@xdef` but the name of the control sequence (without the leading backslash) is used. [§2.1]

`\protected@edef{cs}{arg-syntax}{definition}`

Defined in: L^AT_EX Kernel.

Similar to `\edef` but a protected expansion is performed on `<definition>`. [§2.1]

`\protected@write{output-stream}{init-code}{text}`

Defined in: L^AT_EX Kernel.

Writes `<text>` to the file identified by `<output stream>`. [§9.4]

`\protected@xdef{cs}{arg-syntax}{definition}`

Defined in: L^AT_EX Kernel.

Similar to `\xdef` but a protected expansion is performed on `<definition>`. [§2.1]

`\providecommand{cmd}{n-args}{default}{text}`

Defined in: L^AT_EX Kernel.

Defines the command only if it doesn't already exist. [§2.1]

`\ProvidesClass{name}{version}`

Defined in: L^AT_EX Kernel.

Identifies the class name and optionally the version date. [§11.1]

`\ProvidesPackage{name}{version}`

Defined in: L^AT_EX Kernel.

Identifies the package name and optionally the version date. [§4.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\PRphone{\langle number\rangle}

Defined in: pressrelease class.

Specifies the company's phone number. [§6.2]

\PRsubheadline{\langle text\rangle}

Defined in: pressrelease class.

Specifies the sub-headline. [§6.2]

\PRurl{\langle website\rangle}

Defined in: pressrelease class.

Specifies the company's website. [§6.2]

\ps

Defined in: Classes that define the `letter` environment.

Indicates the start of the postscript. [§3.1]

\psbarcode[\langle options\rangle]{\langle text or filename\rangle}{\langle PS options\rangle}{\langle type\rangle}

Defined in: pst-barcode package.

Generates a bar code. [§10.4]

\psitem{\langle text\rangle}

Defined in: newlfm class.

Specifies the PS line. [§3.3]

\PSNrandom{\langle register\rangle}{\langle n\rangle}

Defined in: probsoln package.

Randomly generates an integer between 1 and $\langle n \rangle$ (inclusive) and stores the result in the given register. [§9.3]

\PSNrandseed{\langle n\rangle}

Defined in: probsoln package.

Sets the random number generator seed. [§9.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\begin{pspicture}[\langle baseline\rangle] (\llx,\lly) (\urx,\ury)`

Defined in: pstricks package.

Environment for drawing vector graphics. [§10.4]

`\PushButton[\langle options\rangle]{\langle label\rangle}`

Defined in: hyperref package.

A push button that performs some action (for use within the Form environment.) [§11.2]

`\put(\langle x\rangle,\langle y\rangle){\langle object\rangle}`

Defined in: L^AT_EX Kernel.

For use inside the picture environment, this puts `\langle object\rangle` at the given co-ordinates (which are in terms of \unitlength). [§10.1]

Q

`\qbezier[\langle points\rangle](\langle start-`

`\rangle,\langle start-y\rangle)(\langle control-x\rangle,\langle control-y\rangle)(\langle end-x\rangle,\langle end-y\rangle)`

Defined in: L^AT_EX Kernel.

For use in the picture, this draws a quadratic Bézier curve with the given start, end and curvature control points. [§10.1]

`\qquad`

Defined in: L^AT_EX Kernel.

Horizontal spacing command (twice as wide as \quad). [§11.1]

`\quad`

Defined in: L^AT_EX Kernel.

Horizontal spacing command equal to the current font's em value.

`\begin{question}[\langle options\rangle]\langle points\rangle`

Defined in: exsheets package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Creates a new question. Both arguments are optional! [§9.2]

`\question[⟨points⟩]`

Defined in: exam class.

Starts a new numbered question, with optionally the number of points the question is worth. [§9.1]

`\begin{questions}`

Defined in: exam class.

Contains all the question in the exam. [§9.1]

R

`\raggedright`

Defined in: L^AT_EX Kernel.

Ragged-right paragraph justification. [§4.3]

`\random{⟨counter⟩}{⟨min⟩}{⟨max⟩}`

Defined in: probsoln package.

Randomly generates an integer between $⟨min⟩$ and $⟨max⟩$ (inclusive) and stores the result in the given counter. [§9.3]

`\re{⟨text⟩}`

Defined in: newlfm class.

Specifies the “re” part of a memo. [§6.1]

`\ref{⟨string⟩}`

Defined in: L^AT_EX Kernel.

References the value of the counter linked to the given label. A second (possibly third) run of L^AT_EX is required to ensure the cross-references are up-to-date. (See **Volume 1** [93, §5.5].) [§1.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\refstepcounter{<counter>}`

Defined in: L^AT_EX Kernel.

Increments the value of the given counter by one and allows the counter to be cross-referenced using `\ref` and `\label`. [§2.1]

`\regarding{<text>}`

Defined in: newlfm class.

Specifies subject of letter. [§3.3]

`\relax`

Defined in: T_EX primitive.

Unexpandable nothing. [§2.1]

`\release{<text>}`

Defined in: newlfm class.

Specifies the release information. [§6.2]

`\renewcommand{<cmd>}[<n-args>]
[<default>]{<text>}`

Defined in: L^AT_EX Kernel.

Redefines an existing command.
(See **Volume 1** [93, §8.2].) [§2.1]

`\renewenvironment{<env-name>}
[<n-args>][<default>]{<begin-
code>}{<end-code>}`

Defined in: L^AT_EX Kernel.

Redefines an existing environment. [§6.5]

`\replaced[<options>]{<new-text>}{<old-text>}`

Defined in: changes package.

Indicates that *<old-text>* has been replaced with *<new text>*. [§13.1]

`\RequirePackage[<options>]
{<name>}[<version>]`

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Analogous to `\usepackage` but for use in class or package files. [§4.2]

`\Reset[⟨options⟩]{⟨label⟩}`

Defined in: hyperref package.

A reset button that resets the form (for use within the `Form` environment.) [§11.2]

`\resizebox{⟨h-length⟩}{⟨v-length⟩}{⟨text⟩}`

Defined in: graphicx package.

Scales the specified contents to the given dimensions. [§10.3]

`\RestartCensoring`

Defined in: censor package.

Switches on the censoring. [§6.4]

`\result`

Defined in: minutes package.

For use within the `Argumentation` environment, this indicates the start of an item indicating the result of the argument. [§6.3]

`\returnaddress`

Defined in: envlab package.

The return address. Defaults to the address given by `\address` [§3.6]

`\rightarrow`

Defined in: L^AT_EX Kernel (Math Mode).

Right arrow →. [§9.3]

`\rlap{⟨text⟩}`

Defined in: L^AT_EX Kernel.

Places `⟨text⟩` to the right of the reference point without taking up any space. [§11.0]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\rmfamily

Defined in: L^AT_EX Kernel.

Switches to the predefined serif font. (Defaults to Computer Modern Roman.)

\Roman{\langle counter\rangle}

Defined in: L^AT_EX Kernel.

Displays counter value as an upper case Roman number. (I, II, III, ...) [§9.2]

\roman{\langle counter\rangle}

Defined in: L^AT_EX Kernel.

Displays counter value as a lower case Roman number. (i, ii, iii, ...) [§6.5]

\romannumeral{\langle number\rangle}

Defined in: T_EX primitive.

Expands $\langle number \rangle$ to a lower case Roman numeral. [§6.5]

\rotatebox[\langle option-list\rangle]{\langle angle\rangle}{\langle text\rangle}

Defined in: graphicx package.

Rotates the given contents by the given angle. [§3.6]

\rule[\langle raise\rangle]{\langle width\rangle}{\langle height\rangle}

Defined in: L^AT_EX Kernel.

Draws a rule (filled rectangle) with the given width and height. If the optional argument is present, the rule is raised by that amount. [§3.6]

S

\sb{\langle maths\rangle}

Defined in: L^AT_EX Kernel (Math Mode).

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Displays its argument as a subscript.

`\begin{scope}[\langle options \rangle]`

Defined in: tikz package.

For use within the `tikzpicture` environment, this provides a way of scoping options. [§7.5]

`\scshape`

Defined in: L^AT_EX Kernel.

Switches to the small-caps form of the current font family, if it exists. See Volume 1 [93, §4.5.1]. [§10.2]

`\begin{Secret}`

Defined in: minutes package.

The environment contents will only be displayed if the package option `Secret` is used. [§6.3]

`\secret{\langle secret text \rangle}`

Defined in: minutes package.

The `\langle secret text \rangle` will only be displayed if the package option `Secret` is used. [§6.3]

`\sectfont`

Defined in: leaflet class.

The font declaration used by the sectioning commands. [§10.3]

`\section[\langle short-title \rangle]{\langle title \rangle}`

Defined in: Most classes that have the concept of document structure.

Inserts a section header. [§1.2]

`\selectfont`

Defined in: L^AT_EX Kernel.

Used to select the current font after the font attributes have been

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

set via commands like
`\fontfamily.`

`\setbeamercovered{<options>}`

Defined in: beamer class.

Sets how covered material should appear. [§8.1]

`\SetBigLabel{<width>} {<height>} {<top>} {<left>} {<sep>} {<columns>} {<rows>}`

Defined in: enlab package.

Sets the custom dimensions for the big labels. [§3.6]

`\setbool{<name>} {<state>}`

Defined in: etoolbox package.

Sets the state (true or false) of a boolean variable. (Local effect only.) A new boolean variable can be defined using `\newbool` and can

be tested in the first argument of `\ifbool`. [§9.4]

`\setboolean{<name>} {<state>}`

Defined in: ifthen package.

Sets the state (true or false) of a boolean variable. (Local effect only.) A new boolean variable can be defined using `\newboolean` and can be tested in the first argument of `\ifthenelse` with `\boolean`. [§6.1]

`\setcommittee{<committee name>}`

Defined in: meetingmins class.

Sets the committee name. [§6.3]

`\setcounter{<counter>} {<number>}`

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Sets the value of a counter. [§10.5]	Sets the options provided by the given option family [§9.3]	
<code>\setdate{<date>}</code>	<code>\setkomavar{<name>}{<description>}{<content>}</code>	
Defined in: meetingmins class.	Defined in: KOMA-Script classes.	
Sets the date of the meeting. [§6.3]	Sets the content and optionally the description of a KOMA-Script variable. [§3.2]	
<code>\setdynamiccontents{<id>}{<contents>}</code>	<code>\setkomavar*{<name>}{<description>}</code>	
Defined in: flowfram package.	Defined in: KOMA-Script classes.	
Sets the contents of a dynamic frame. [§10.5]	Sets the description of a KOMA-Script variable. [§3.2]	
<code>\SetEnvelope[<top-margin>]{<width>}{<height>}</code>	<code>\SetLabel{<width>}{<height>}{<top>}{<left>}{<sep>}{<columns>}{<rows>}</code>	
Defined in: envlab package.	Defined in: envlab package.	
Sets the custom envelope dimensions. [§3.6]	Sets the custom label dimensions. [§3.6]	
<code>\setkeys{<family>}{{<options>}}</code>		
Defined in: keyval package.		

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\setlength{\<register>}{\<dimension>}`

Defined in: L^AT_EX Kernel.

Sets the value of a length register.

[§6.5]

`\setmargins{\<top>}{\<bottom>}{\<left>}{\<right>}`

Defined in: leaflet class.

Used to specify the page margins.

[§10.3]

`\setmembers{\<list>}`

Defined in: meetingmins class.

Sets the list of members. The chair should be indicated within the list with the `\chair` command.
[§6.3]

`\setpresent{\<list>}`

Defined in: meetingmins class.

Sets the list of people present. The chair should be indicated within the list with the `\chair` command.
[§6.3]

`\setsocextension{\<extension>}`

Defined in: changes package.

Sets the extension of the list or summary of changes file. [§13.1]

`\setstaticcontents{\<id>}{\<contents>}`

Defined in: flowfram package.

Sets the contents of a static frame.
[§10.5]

`\setupdocument{\<options>}`

Defined in: isodoc class.

Sets up the general document options [§3.4]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\SetupExSheets[⟨module⟩]
{⟨options⟩}`

Defined in: exsheets package.
Sets options for the exsheet package. [§9.2]

`\sffamily`

Defined in: L^AT_EX Kernel.
Switches to the predefined sans-serif font. (Defaults to Computer Modern Sans.) [§12.3]

`\shortstack[⟨align⟩]{⟨text⟩}`

Defined in: L^AT_EX Kernel.
Produces a box with a single column of text with the reference point at the lower-left corner.
Similar to using a single column `tabular` environment. [§10.1]

`\show⟨token⟩`

Defined in: T_EX primitive.

Interrupts the document compilation and writes the definition of *⟨token⟩* to the transcript. [§2.1]

`\showanswers`

Defined in: probsoln package.
Shows the solutions (from that point onwards). [§9.3]

`\signature{⟨text⟩}`

Defined in: letter class; some other letter-like classes; minutes package.

Specifies the sender's (or author's) name to go after the closing text. [§3.1]

`\sin`

Defined in: L^AT_EX Kernel (Math Mode).

Typesets sin function name. [§9.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\small`

Defined in: Most document classes.

Switches to small sized text. [§6.4]

`\smallskip`

Defined in: L^AT_EX Kernel.

Inserts a small vertical space. The size is given by the length `\smallskipamount`. [§12.3]

`\begin{solution}[(options)]`

Defined in: exsheets package.

Contains the solution to the preceding question. [§9.2]

`\begin{solution}`

Defined in: probsoln package.

Starts a new paragraph (without indentation), typesets

`\solutionname` in bold followed by a colon space and then the environment contents. [§9.3]

`\begin{solution}[(length)]`

Defined in: exam class.

Typesets its contents if the `answers` option is set otherwise it hides its contents, optionally replacing the contents with space for the student to fill in their answer. [§9.1]

`\solutionname`

Defined in: probsoln package.

The solution title text used by the `solution` environment. [§9.3]

`\begin{solutionorbox}[(length)]`

Defined in: exam class.

Typesets its contents if the `answers` option is set otherwise it

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

hides its contents, optionally replacing the contents with an empty box for the student to fill in their answer. [§9.1]

```
\begin{solutionordottedlines}  
[(length)]
```

Defined in: exam class.

Typesets its contents if the answers option is set otherwise it hides its contents, optionally replacing the contents with an area containing dotted lines for the student to fill in their answer. [§9.1]

```
\begin{solutionorlines}  
[(length)]
```

Defined in: exam class.

Typesets its contents if the answers option is set otherwise it hides its contents, optionally replacing the

contents with a lined area for the student to fill in their answer. [§9.1]

```
\sp{\langle maths \rangle}
```

Defined in: L^AT_EX Kernel (Math Mode).

Displays its argument as a superscript.

```
\space
```

Defined in: L^AT_EX Kernel.

A space character. Sometimes used instead of typing a space character when the author wants to emphasize that there's supposed to be a space there. For example, if the space occurs at the end of a line of code, using \space is a visual indication that the author hasn't simply forgotten to discard an unwanted EOL character. [§3.6]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\Square

Defined in: wasysym package or ifsym with the geometry option.

Produces a square symbol \square .

[§10.3]

\startlabels

Defined in: enlab package.

Starts the manual generation of labels (which should be made using \mlabel). [§3.6]

\starttime{\langle time\rangle}

Defined in: minutes package.

Specifies the start time of the meeting. [§6.3]

\begin{staticcontents}{\langle id\rangle}

Defined in: flowfram package.

Sets the contents of a static frame. [§10.5]

\STExpenses

Defined in: invoice package.

For use within the `invoice` environment, this command is used to make a subtotal appear for all the expenses hidden via \EBCi or \EFCi. [§4.2]

\StopCensoring

Defined in: censor package.

Switches off the redaction to produce an uncensored version of the document. [§6.4]

\string(cs)

Defined in: T_EX primitive.

Converts the given control sequence into the list of characters that makes up that control sequence name (including the initial backslash) where each

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

character in the list has category code 12 ("other") [§11.1]

`\begin{subitems}`

Defined in: meetingmins class.

A numbered list for use within subsections and sub-subsections.

Use the `items` environment for lists within sections. [§6.3]

`\Submit[<options>]{<label>}`

Defined in: hyperref package.

A submit button that sends the data to the URL provided by the form's action (for use within the `Form` environment.) [§11.2]

`\subparagraph[<short-title>]{<title>}`

Defined in: Most classes that have the concept of document structure.

Inserts a subsubsubsubsection header. Most classes default to an unnumbered running header for this sectional unit. [§6.5]

`\subpart[<points>]`

Defined in: exam class.

Starts a new numbered question sub-part, with optionally the number of points the sub-part is worth. [§9.1]

`\begin{subparts}`

Defined in: exam class.

Contains all the question sub-parts. [§9.1]

`\subsection[<short-title>]{<title>}`

Defined in: Most classes that have the concept of document structure.

Inserts a subsection header. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\subsubpart[⟨points⟩]`

Defined in: exam class.

Starts a new numbered question sub-sub-part, with optionally the number of points the sub-sub-part is worth. [§9.1]

`\begin{subsubparts}`

Defined in: exam class.

Contains all the question sub-sub-parts. [§9.1]

`\subsubsection[⟨short-title⟩]{⟨title⟩}`

Defined in: Most classes that have the concept of document structure.

Inserts a subsubsection header.

[§6.5]

`\subtitle{⟨title⟩}`

Defined in: Various classes or

packages that have the concept of a subtitle.

Specifies the subtitle. Usually for use with `\maketitle` in a similar manner to `\title`. Some classes, such as beamer, also provide an optional argument for this command. [§6.3]

`\subtopic[⟨toc title⟩]{⟨title⟩}`

Defined in: minutes package.

Starts a new subtopic. [§6.3]

`\svnId`

Defined in: svninfo package.

Prints a summary of the Subversion information in the same form as the `Id` keyword anchor. [§13.2]

`\svnInfo$⟨Id⟩$`

Defined in: svninfo package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Parses the Subversion Id keyword anchor. [§13.2]	\svnInfoHeadURL	Symbols
\svnInfoDate	Defined in: svninfo package.	A N
Prints the date in the form <code><YYYY>-<MM>-<DD></code> when the file was checked out or the current date if unknown. [§13.2]	Prints the information obtained from the HeadURL keyword. [§13.2]	B O
\svnInfoDay	\svnInfoLongDate	C P
Defined in: svninfo package.	Defined in: svninfo package.	D Q
Prints the day (in <code><DD></code> form) when the file was checked out or the current day if unknown. [§13.2]	Prints date in the form of \today when the file was checked out or the current date if unknown. [§13.2]	E R
\svnInfoFile	\svnInfoMaxRevision	F S
Defined in: svninfo package.	Defined in: svninfo package.	G T
Prints the name of the source file or --sourcefile-- if unknown. [§13.2]	Prints the maximum revision number for all the files in the document or --maxrevision-- if unknown. [§13.2]	H U
	\svnInfoMaxToday	I V
	Defined in: svninfo package.	J W
		K X
		L Y
		M Z

Summary of Commands and Environments

Prints date in the form of `\today` from the latest Subversion revision. [§13.2]

`\svnInfoMinRevision`

Defined in: `svninfo` package.

Prints the minimum revision number for all the files in the document or `--minrevision--` if unknown. [§13.2]

`\svnInfoMonth`

Defined in: `svninfo` package.

Prints the month (in `<MM>` form) when the file was checked out or the current month if unknown. [§13.2]

`\svnInfoOwner`

Defined in: `svninfo` package.

Prints the user name of the file owner or `--owner--` if unknown. [§13.2]

`\svnInfoRevision`

Defined in: `svninfo` package.

Prints the revision number of the checked out file or `--revision--` if unknown. [§13.2]

`\svnInfoTime`

Defined in: `svninfo` package.

Prints the time when the file was checked out or `--time--` if unknown. [§13.2]

`\svnInfoYear`

Defined in: `svninfo` package.

Prints the year when the file was checked out or the current year if unknown. [§13.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\svnKeyword{\$<keyword>\$}

Defined in: `svninfo` package.

Parses the Subversion keyword anchor. [§13.2]

T

\begin{table}[\langle placement\rangle]

Defined in: Most classes except for those designed for correspondence or similarly restrictive documents.

Floats the contents to the nearest location according to the preferred placement options, if possible. Within the environment, `\caption` may be used one or more times, as required. (See Volume 1 [93, §7.2].) [§2.6]

\tableofcontents

Defined in: Most classes that have the concept of document structure.

Inserts the table of contents. A second (possibly third) run is required to ensure the page numbering is correct. [§6.3]

\begin{tabular}[\langle v-pos\rangle]{\langle col-specs\rangle}

Defined in: L^AT_EX Kernel (Text Mode).

Environment for lining things up in rows and columns. [§2.6]

\task[\langle footnote-text\rangle]{\langle name\rangle}{\langle when\rangle}{\langle text\rangle}

Defined in: `minutes` package.

Specifies a task. [§6.3]

\task*[\langle when\rangle]{\langle text\rangle}

Defined in: `minutes` package.

Specifies a task. [§6.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\telephone{<text>}

Defined in: letter class.

Specifies the sender's telephone number. [§3.1]

\TeX

Defined in: L^AT_EX Kernel.

Typesets the T_EX logo. [§7.4]

\text{<text>}

Defined in: amsmath package (Math Mode).

Displays its argument in the normal text font (as opposed to the current maths font). [§7.4]

\textasciicircum

Defined in: L^AT_EX Kernel.

Circumflex ^ symbol. [§2.2]

\textasciitilde

Defined in: L^AT_EX Kernel.

Tilde ~ symbol. (If you are typing an URL, use the url package, which provides \url{<address>} that allows you to directly type ~ in the address.) [§2.2]

\textbf{<text>}

Defined in: L^AT_EX Kernel.

Renders <text> with a bold weight in the current font family, if it exists. (See Volume 1 [93, §4.5.1].) [§1.0]

\textcolor[<model>]{<specs>}{<text>}

Defined in: color and xcolor packages.

Sets <text> with the foreground colour according to the given <specs>. [§2.3]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\textdollar

Defined in: textcomp package.

Displays the dollar symbol \$. [§4.2]

\begin{textenum}

Defined in: probsoln package.

An inline numbered list environment. Each item may be started with the standard \item command, but may also be started with \correctitem or \incorrectitem. [§9.3]

\texteuro

Defined in: textcomp package.

Displays the Euro symbol €. [§4.2]

\TextField[*options*]{*label*}

Defined in: hyperref package.

A text field (for use within the Form environment.) [§11.2]

\textheight

Defined in: L^AT_EX Kernel.

A length containing the height of the typeblock. Note that the actual contents of the page may fall short of the text height (underfull vbox) or extend beyond it (overfull vbox). This measurement does not include the header and footer areas. [§10.5]

\textifsymbol[*font-family*]{*number*}

Defined in: ifsym package.

Selects the symbol identified by *number* from the given font family. [§11.1]

\textsubscript{*text*}

Defined in: fixltx2e package (now in L^AT_EX Kernel).

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Makes $\langle text \rangle$ a subscript. Unlike `\sb`, this command is for use in text mode. [§13.1]

`\langle text \rangle`

Defined in: L^AT_EX Kernel.

Makes $\langle text \rangle$ a superscript. Unlike `\sp`, this command is for use in text mode. [§13.1]

`\texttt{\langle text \rangle}`

Defined in: L^AT_EX Kernel.

Renders $\langle text \rangle$ in the predefined monospaced font. See

Volume 1 [93, §4.5.1]. [§2.1]

`\textwidth`

Defined in: L^AT_EX Kernel.

A length containing the width of the typeblock. Note that the actual contents of the line may fall short

of the line width (underfull hbox) or extend beyond it (overfull hbox). This width does not include the area for marginal notes. [§2.1]

`\the\langle register \rangle`

Defined in: T_EX primitive.

Expands $\langle register \rangle$ to the current value of the register. [§2.1]

`\begin{thebibliography}\{\langle widest entry label \rangle\}`

Defined in: Most classes that define sectioning commands.

Bibliographic list. (See also `\bibitem` and `\cite`).

`\theenumi`

Defined in: L^AT_EX Kernel.

Displays the current value of the enumi (first level enumerate) counter. [§6.5]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\theenumii

Defined in: L^AT_EX Kernel.

Displays the current value of the enumii (second level enumerate) counter. [§6.5]

\theenumiii

Defined in: L^AT_EX Kernel.

Displays the current value of the enumiii (third level enumerate) counter. [§6.5]

\theenumiv

Defined in: L^AT_EX Kernel.

Displays the current value of the enumiv (fourth level enumerate) counter. [§6.5]

\begin{theorem}[\langle title\rangle]

Defined in: beamer class.

An environment for typesetting theorems. [§8.0]

\thequestion

Defined in: exam class.

Displays the current question number. [§9.1]

\thesection

Defined in: L^AT_EX Kernel.

Displays the current value of the section counter [§6.5]

\thicklines

Defined in: L^AT_EX Kernel.

This declaration switches to a thick line width for lines drawn within the `picture` environment. [§10.1]

\thispagestyle{\langle style\rangle}

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Like `\pagestyle` but only affects the current page. [§3.1]

`\thisproblem`

Defined in: probsoln package.

May be used within the `<body>` argument of `\foreachproblem` to display the current problem. [§9.3]

`\thisproblemlabel`

Defined in: probsoln package.

May be used within the `<body>` argument of `\foreachproblem` to access the current problem label. [§9.3]

`\tick`

Defined in: pgfplots package.

For use within one of the tick label options, this expands to the current tick element. [§12.5]

`\ticket{<content>}`

Defined in: ticket package.

Specifies the variable ticket content. [§10.2]

`\ticketdefault`

Defined in: ticket package.

The default ticket content. [§10.2]

`\ticketDistance{<x-dist>} {<y-dist>}`

Defined in: ticket package.

The horizontal and vertical distances between tickets in terms of `\unitlength`. [§10.2]

`\ticketNumbers{<num-cols>} {<num-rows>}`

Defined in: ticket package.

Specifies the number of tickets per sheet and their arrangement. [§10.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\ticketSize{<width>}{'height'}`

Defined in: ticket package.

The width and height of the tickets in terms of `\unitlength`. [§10.2]

`\ticknum`

Defined in: pgfplots package.

For use within one of the tick label options, this expands to the current tick number (starting from 0). [§12.5]

`\begin{tikzpicture}[<options>]`

Defined in: tikz package.

Environment for drawing vector graphics. [§7.5]

`\time`

Defined in: T_EX primitive.

The current time expressed as the number of minutes since midnight. [§7.4]

`\title{<text>}`

Defined in: Most classes that have the concept of a title page.

Specifies the document title. This command doesn't display any text so may be used in the preamble, but if it's not in the preamble it must be placed before `\maketitle`. Some classes, such as beamer, provide an optional argument for this command. [§5.2]

`\titledquestion{<title>}[<points>]`

Defined in: exam class.

Like `\question` but assigns a title to the question. [§9.1]

`\titlegraphic{<graphic>}`

Defined in: beamer class.

Code to produce a title graphic used by `\maketitle`. [§8.0]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

\ToAddressWidth

Defined in: envlab package.

A length register used to store the width of the recipient's address.

[§3.6]

\today

Defined in: Most of the commonly-used classes.

Inserts into the output file the date when the L^AT_EX application created it from the source code. [§3.3]

toFile(<filename>)

Defined in: arara directive.

Create a file reference. [§1.2]

\topic[<toc title>]{<title>}

Defined in: minutes package.

Starts a new topic. [§6.3]

\toprule{<wd>}

Defined in: booktabs package.

Horizontal rule for the top of a **tabular** environment. [§2.6]

\totalpoints

Defined in: exsheets package.

Displays the total number of points, including bonus points. (Requires two L^AT_EX runs to ensure it's up to date.) The starred version omits the unit. [§9.2]

\two@digits{<number>}

Defined in: L^AT_EX Kernel.

Displays <number>, ensuring that it has at least two digits. (If <number> is less than 10 a leading 0 is inserted.) [§7.4]

\twocolumn

Defined in: L^AT_EX Kernel.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Issues a page break and switches to two column mode. [§10.5]

U

`unchanged(<ref>)`

Defined in: arara directive.

Evaluates to true if the given file hasn't changed. The argument `<ref>` may either be a string "`<extension>`" which indicates the file extension or a file reference `toFile("<filename>")`. [§1.2]

`\unframedsolutions`

Defined in: exam class.

Switches off the default frame around the solutions. [§9.1]

`\unitlength`

Defined in: L^AT_EX Kernel.

A length used as the unit of measurement in the `picture` environment. [§10.1]

`unless <condition>`

Defined in: arara directive.

Only run the application if `<condition>` is false. [§1.2]

`until <condition>`

Defined in: arara directive.

Repeatedly run the application until `<condition>` is true. [§1.2]

`\url{<address>}`

Defined in: url package.

Typesets an URL in a typewriter font and allows you to use characters such as ~. [§6.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\usecolortheme[options]{name}`

Defined in: beamer class.

Loads a beamer color theme.

[§8.2]

`\usefonttheme[options]{name}`

Defined in: beamer class.

Loads a beamer font theme. [§8.2]

`\useinnertheme[options]{name}`

Defined in: beamer class.

Loads a beamer inner theme.

[§8.2]

`\useoutertheme[options]{name}`

Defined in: beamer class.

Loads a beamer outer theme.

[§8.2]

`\usepackage[option-list]{package-list}`

Defined in: L^AT_EX Kernel.

Loads the listed package(s). (See
Volume 1 [93, §4.2].) [§1.1]

`\useproblem[dataset]{label}
{arg1}... {argN}`

Defined in: probsoln package.

Uses a previously defined problem.
[§9.3]

`\usetHEME[options]{name}`

Defined in: beamer class.

Loads a beamer presentation
theme. [§8.2]

`\usetikzlibrary{name}`

Defined in: tikz package.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

Load the tikz library called $\langle name \rangle$.
(Preamble only.) [§7.5]

V

`\value{\langle counter \rangle}`

Defined in: L^AT_EX Kernel.

References the value of the given counter where a number rather than a counter name is required.
[§2.1]

`\vector(\langle h \rangle,\langle v \rangle){\langle length \rangle}`

Defined in: L^AT_EX Kernel.

For use within the argument of `\put`, this draws a straight line with an arrowhead of the given length whose horizontal and vertical extent (gradient vector) is given by $(\langle h \rangle,\langle v \rangle)$. [§10.1]

`\verb<char>\langle text \rangle<char>`

Defined in: L^AT_EX Kernel.

Typesets $\langle text \rangle$ verbatim. The starred version replaces spaces with the visible space symbol.
[§8.0]

`\begin{verbatim}`

Defined in: L^AT_EX Kernel.

Typesets the contents of the environment as is. (Can't be used in the argument of a command.)
[§2.3]

`\verbatiminput{\langle filename \rangle}`

Defined in: verbatim package.

Inputs the given file verbatim.
[§9.4]

`\vfill`

Defined in: L^AT_EX Kernel.

Inserts a vertical space that will expand to fit the available height.
[§3.6]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\begin{Vote}`

Defined in: minutes package.

If multiple votes were made, the `\vote` commands are listed in this environment. [§6.3]

`\vote{\langle description \rangle}{\langle yes \rangle}{\langle no \rangle}{\langle abstain \rangle}{[\langle decision \rangle]}`

Defined in: minutes package.

Indicates a vote took place. [§6.3]

`\vspace{\langle length \rangle}`

Defined in: L^AT_EX Kernel.

Inserts a vertical gap of the given height. The unstarred version doesn't create a space if it occurs at the beginning or end of a page. The starred version always creates a space. [§9.1]

W

`while {\langle condition \rangle}`

Defined in: arara directive.

Repeatedly run the application while `\langle condition \rangle` is true. [§1.2]

X

`\xappto{\cs}{\langle code \rangle}`

Defined in: etoolbox package.

Global version of `\eappto`. [§2.1]

`\xblackout{\langle text \rangle}`

Defined in: censor package.

Like `\blackout`, but also blacks out the interword spaces. [§6.4]

`\XBox`

Defined in: wasysym package.

Produces a square with a cross in it \boxtimes . [§11.1]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Summary of Commands and Environments

`\xdef{cs}{arg-syntax}
 {definition}}`

Defined in: TeX primitive.

As `\edef` but the definition is global. [§2.1]

`\xDTAssignfirstmatch{<db-name>}{{<col-label>}}{<value>}
 {<assign-list>}`

Defined in: datatool package.

Finds the first row in the database `<db-name>` where entry in the column identified by the label `<col-label>` matches a *one level expansion* of `<value>` and applies the assignment list, which has the same format as for `\DTLforeach` and `\DTLforeach*`. See also `\DTAssignfirstmatch`. [§2.8]

`\xifinlist{<item>}{{<list-cs>}}`

`{<true>}{{<false>}}`

Defined in: etoolbox package.

As `\ifinlist` but the test is performed on the expansion of `<item>`. [§2.7]

`\xifinlistcs{<item>}{{<list-csname>}}{<true>}{{<false>}}`

Defined in: etoolbox package.

As `\ifinlist` but the control sequence name is supplied (without the backslash). [§2.7]

`\xpreto{cs}{<code>}`

Defined in: etoolbox package.

Global version of `\preto`. [§2.1]

Y

`\year`

Defined in: TeX primitive.

The current year. [§7.2]

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

INDEX

Page numbers in **bold** indicate the entry definition in the summary.

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

Symbols

!	737, 833, 901, 1058	\@afterheading	454, 1062
„	5, 1056, 1057, 1059	\@auxout	698, 1062
“	70	\@currenvir	31, 1062
#	37, 74, 194, 476, 500, 867, 1055, 1059	\@endfortrue	1063
##	192, 194, 1059	\@enumdepth	447, 1062
\$	74, 75, 747, 1051, 1055, 1059	\@firstoftwo	846, 1062
%	5, 74, 106, 643, 1027, 1051, 1054, 1055, 1059	\@for	163, 168, 170, 1051, 1054, 1062
% arara:	13, 1060	\@genderlist	858
&	74, 153, 308, 322, 357, 1055, 1060	\@ifundefined	47, 1063
--	57, 350, 386, 905, 1060, 1061	\@listdepth	448
---	216, 1061	\@namedef	48, 1063
<	906, 1061	\@nameuse	47, 1063
>	330, 906, 1061	\@projectlist	858
\@	1051, 1062	\@secondoftwo	847, 1063
@	740, 1051, 1054, 1061, 1145, 1146	\@timezonefmt	507
		\@toodeep	447, 1064
		[5, 1064

\& 74, 592, 1064
 \\$ 74, 314, 1064
 # 74, 1064
 % 74, 523, 1064
 \ 1055, 1064
 \" 398, 1064
 \, 330, 1064
 \[668, 705, 1065
 \\ 308, 322, 328, 357, 715, 953, 956, 1065
 \` 337, 721, 877, 1065
 \^ 510, 1065
 _ 668, 705, 1065
 \` 74, 1065
 \` 330, 1066
 `` 162, 1066
 ' 330, 1060
 '' 162, 1060
 \{ 74, 1065
 \} 74, 1065
] 5, 1066
 ^ 74, 1055, 1066, 1140
 - 74, 1055, 1066, 1140
 { 3, 74, 159, 1055, 1066
 } 3, 74, 1055, 1067
 ~ 74, 1050, 1056, 1067

A

about environment 386, 1067
 accept (isodoc) 307
 acceptaccount (isodoc) 307
 acceptaddress (isodoc) 307
 acceptcents (isodoc) 307
 acceptdescription (isodoc) 307
 accepteuro (isodoc) 307
 acceptreference (isodoc) 308
 \accountdata 308, 310, 1067
 accountname (isodoc) 307
 accountno (isodoc) 306
 active character 1050, 1056
 \added 982, 987, 1067
 id 982
 remark 982
 addedmarkup (changes) 984
 \addevent 564
 \addplot 962, 963, 965, 971, 1067
 \addpoints 608, 1067
 addpoints (exam) 608, 609, 614, 618
 \address 226, 271, 284, 285, 295, 1067
 addrfield (KOMA option) 238
 \addrfrom 252, 274, 300, 377, 381, 1068
 addrfromemail (newlfm) 255
 addrfromfax (newlfm) 255

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

		Symbols
		A N
addrfromleft (newlfm)	255	
addrfromphone (newlfm)	255	
addrfromright (newlfm)	255	
\addrto	251, 276, 301, 1068	
addrtoemail (newlfm)	255	
addrtofax (newlfm)	255	
addrtoleft (newlfm)	255	
addrtophone (newlfm)	255	
addrtright (newlfm)	255	
\AddToBackground	734, 735, 737, 746, 1068	
\addtolength	445, 1068	
Adobe Reader	461, 871, 878	
\advance	59, 1068	
agenda (meetingmins)	388, 389, 392	
Alegreya package	735	
\Alph	438, 636, 647, 1068	
\alph	438, 636, 1068	
amsmath package	513, 866	
\and	572, 592, 1069	
answers (exam)	609, 619, 625	
answers (probsoln)	638, 640, 658, 659	
\appenddynamiccontents	763, 1069	
\appto	48, 564, 856, 857, 1069	
\arabic	438, 440, 636, 1069	
arara	12–20, 94, 102, 104, 109, 750, 756,	
arara directives	1033, 1050	
!	14, 1058	
&	20, 1060	
&&	14, 20, 1060	
-->	17, 1061	
	20, 1066	
	14, 20, 99, 1066	
changed	14, 15, 17, 19, 20, 99, 1076	
datatooltk	94	
dvips	750	
exists	15, 1118	
found	15, 1121	
if	14, 99, 1128	
latex	750	
missing	14, 15, 17, 99, 1148	
pdflatex	13, 15, 18	
ps2pdf	750	
toFile	14, 99, 1197	
unchanged	15, 20, 1198	
unless	14, 1198	
until	14, 1198	
while	14, 1201	
xelatex	292	
areacode (isodoc)	262	
Argumentation environment	404, 1069	

				Symbols
array package	329	\backgroundsetup	423, 424, 426, 1070	A N
article class	220, 732, 835	backside (leaflet)	733	B O
\AtBeginDocument	494, 497, 780, 1069	bardiag package	932	C P
\author	107, 572, 1069	\baselineskip	288, 415, 888, 1070	D Q
authormarkup (changes)	985	basic Latin character	1050, 1053	E R
authormarkupposition (changes)	985	\bcbar	935, 1070	F S
authormarkuptext (changes)	985	color	936	G T
autograph (isodoc)	264, 268	label	936	H U
auxiliary file (.aux)	698	plain	936	I V
avery5160label (envlab)	283	text	936	J W
avery5161label (envlab)	283	value	936	K X
avery5162label (envlab)	283	\bcfontstyle	935, 1070	L Y
avery5163biglabel (envlab)	284	bchart environment	934, 936, 1070	M Z
avery5163label (envlab)	283	max	934	
avery5164biglabel (envlab)	284	min	934	
avery5164label (envlab)	283	plain	934	
avery5262label (envlab)	283	scale	935	
axis environment	962, 1070	step	934	
B				
babel package	224, 343, 460, 464, 468, 482, 486, 491, 1050	steps	934	
british	224	unit	934, 939	
english	394	width	935	
backaddress (KOMA option)	238	bchart package	932, 934	
background package	423	\bclabel	937, 1070	
		\bcskip	936, 1071	
		label	936	
		\bc xlabel	937, 1071	

Index

			Symbols
beamer class	570	plain	A N
beamer themes		bibunits package	B O
Boadilla	594	bic (isodoc)	C P
color	590	\BigCircle	D Q
EastLansing	594	\bigskip	E R
font	590	bizcard package	F S
Goettingen	594, 603	\blackout	G T
inner	591	block environment	H U
Montpellier	594	\bonuspart	I V
outer	591	\bonusquestion	J W
presentation	590	\bonussubpart	K X
rounded	603	\bonussubsubpart	L Y
spruce	603	\bonusum	M Z
\begin	31, 446, 764, 865, 1071	\bonustitledquestion	
\bfseries	735, 1071	bookletenvlope (envlab)	
\BgThisPage	423, 1071	booktabs (europecv)	
\bibentry	354, 355, 1071	booktabs package	
bibentry package	353	boolean	
biber	339	\boolean	641, 675, 1073
\bibitem	357, 359, 1072	boolean key	76, 1050
\bibliography	19, 339, 351, 353, 356, 1072	\boolfalse	675, 1073
\bibliographystyle	19, 339, 353, 354, 1072	\booltrue	675, 1074
bibtex	18, 979	bothsides (leaflet)	733
BibTeX styles (.bst)		\bottomrule	148, 1074
		\bracketedpoints	625, 1074
		\breakforeach	1074

			Symbols
british (babel)	224	chair (meetingmins)	A N
businessenvelope (envlab)	282	changes package	B O
busletter (newlfm)	254	addedmarkup	C P
busletternofrom (newlfm)	254	authormarkup	D Q
C		authormarkupposition	E R
c5envelope (envlab)	282	authormarkuptext	F S
c65envelope (envlab)	282	deletedmarkup	G T
c6envelope (envlab)	282	draft	H U
\calendarmonth	565	final	I V
capaddress (envlab)	291, 292	markup	J W
\caption	146, 323, 1074	ulem	K X
category code	71, 106, 468, 501, 1051, 1053–1055, 1057	xcolor	L Y
\cc	222, 223, 228, 247, 273, 286, 297, 396, 398, 1074	\chapter	M Z
\cclist	254, 275, 300, 1075	\CheckBox	
cellphone (isodoc)	265	checkboxes environment	
\censor	410, 1075	\CheckedBox	
censor package	410	\choice	
\censor*	415, 1075	\ChoiceMenu	
\censorbox	411, 1075	choices environment	
\censorbox*	415, 1075	\circle	
center environment	220, 619, 736, 1075	\cite	
\centering	146, 1076	city (isodoc)	
\chair	389, 1076	cityzip (isodoc)	
		class file options (general)	
		10pt	382, 608
		11pt	382, 608

			Symbols
12pt	384, 608, 777, 835	\closing	A N
a4paper	221, 341, 384, 732, 835	closing (isodoc)	B O
letterpaper	5, 384, 732	collect package	C P
onecolumn	758	\Collect@Body	D Q
twocolumn	758	\collect@body	E R
class files (.cls)		collectinmacro environment	F S
article	220, 732, 835	\color	G T
beamer	570	color (databar)	H U
dinbrief	219	color (datapie)	I V
europecv	335, 340, 368	\colorbox	J W
exam	606, 608, 644, 646, 679	column	K X
isodoc	220, 260, 268, 278, 305, 306, 321	header (title)	L Y
leaflet	709, 730, 733, 734	index (<col-idx>)	M Z
letter	219, 220, 268, 278, 284	label (<col-label>)	
meetingmins	387, 388, 394, 402	combine (leaflet)	
newlfm	220, 249, 256, 268, 298, 374, 378	comma-separated list	
pressrelease	382, 387	149, 156, 167, 168, 186, 664, 730, 844, 1051, 1054	
scrartcl	777	comma-separated variable file (.csv)	
scrbook	5	21, 69–73, 75–81, 83, 85, 87, 98, 112, 113, 115, 117, 123, 127, 144, 209, 313, 320	
scrltr2	219, 227, 268	company (isodoc)	
standalone	755	\computeleftedgeeven	
\ClassError	829, 1078	\computeleftedgeodd	
\clearpage	1078	\Contra	
\closeline	250, 1078	\contra	

			Symbols
copyto (isodoc)	264	\csgpreto	A N
\CorrectChoice	622, 624, 1080	\cslet	B O
\correctitem	646, 1080	\csletcs	C P
\correctitemformat	646, 1080	\csname	D Q
\cos	615, 1080	\cspreto	E R
counters		\csuse	F S
DTLbarroundvar	943	CSV 71–73, 75–81, 85, 87, 209, 215, 312,	G T
DTLbrow	367	333, 730, 1051	H U
DTLpieroundvar	918	\csvloop	I V
enumi	437, 645, 1193	\csxappto	J W
enumii	437, 645, 1194	\csxdef	K X
enumiii	437, 1194	\csxpreto	L Y
enumiv	437, 1194	CTAN	M Z
page	453, 761	currency (isodoc)	
seignumdepth	767	\CurrentOption	
section	1194	curriculum vitæ	
country (isodoc)	262	currvita package	
countrycode (isodoc)	261	LabelsAligned	
coverpages environment	619, 1080	ManyBibs	
\csappto	54, 564, 1080	NoDate	
\csdef	43, 549, 564, 829, 1081	openbib	
\cseappto	54, 1081	TextAligned	
\csedef	44, 1081	customer (KOMA variable)	
\csepreno	54, 1081	\CutLine	
\csgappto	54, 1081	\cutline	
\csgdef	44, 1082	CV	
		334–337, 339, 342, 349, 1052	

		Symbols
cv environment	336, 337, 1084	A N
cvlist environment	337, 1084	B O
\cvplace	336, 337, 1084	C P
D		D Q
\dashbox	718, 1084	E R
databar package	932, 940	F S
color	940	G T
gray	940	H U
horizontal	940	I V
vertical	940	J W
database label (< <i>db-name</i> >)	70	K X
databib package	355, 358	L Y
datapie package	913	M Z
color	913	
gray	913	
norotateinner	913	
norotateouter	914	
rotateinner	913	
rotateouter	914	
dataplot package	960	
dataool file (.dbtex)	69, 76, 82, 85, 87, 93, 104, 207, 210	
dataool package	22, 44, 62, 70, 73, 83, 111, 112, 147, 149, 161, 166, 196, 209, 268, 305, 355, 607, 671, 695,	
	932, 940, 1052, 1100	N
	math=pgfmath	O
	datatooltk	P
	22, 76, 83, 85–87, 91–93, 102–105, 112–114, 123, 124, 135, 145, 607, 671, 679, 686, 687, 691, 1052, 1052	Q
	--filter	R
	--filter-and	S
	--filter-exclude	T
	--merge	U
	--seed	V
	--shuffle	W
	--shuffle-iterations	X
	--truncate	Y
	datatooltk-gui	Z
	87, 92, 93, 142, 1052	
	\date	A
	336, 337, 384, 572, 592, 824, 1084	B
	date (KOMA variable)	C
	date (isodoc)	D
	\date(<i>language</i>)	E
	\datebritish	F
	datecenter (newlfm)	G
	\datefmt	H
	483, 486, 489	I
	dateleft (newlfm)	J
	dateno (newlfm)	K
	dateright (newlfm)	L
	\dateset	M
	250, 377, 1085	N

			Symbols
datetime package	460, 499	\DefaultWidthofChoiceMenu	888, 1087
datetime2 package	460, 462, 499, 521	\DefaultWidthofReset	888, 1087
showdow	465	\DefaultWidthofSubmit	887, 1087
useregional	464	\DefaultWidthofText	889, 1087
datetime2-calc package	521	\definechangesauthor	981, 986, 1088
datetime2-english package	465	color	982
dateyes (newlfm)	256	name	981
\day	470, 482, 498, 501, 1085	defproblem environment	650, 1088
\DBIBcitekey	357, 361, 365, 1085	\deleted	982, 987, 1088
\DBIBentrytype	358, 1085	id	982
\decision	408, 1085	remark	982
\decision*	408, 1085	deletedmarkup (changes)	984
\decisionontheme	407, 1086	\Delta	669, 1088
\DeclareOption	821, 1086	\descfont	735, 737, 1088
\DeclareOption*	821, 835, 868, 1086	description environment	735, 1088
\def	37–41, 43, 476, 500, 994, 1086	\dimexpr	57, 767, 768, 815–817, 1088
\DefaultHeightofCheckBox	888, 891, 1086	dinbrief class	219
		\ding	213, 216, 814–817, 878, 1088
\DefaultHeightofChoiceMenu	888, 1086	dinglist environment	767, 1089
\DefaultHeightofReset	887, 1086	disablegeometry (xwatermark)	429
\DefaultHeightofSubmit	887, 1087	\Discount	317, 1089
\DefaultHeightofText	888, 1087	\divide	61, 479, 1089
\DefaultHeightofTextMultiline	888, 1087	dlenvelope (enlab)	282
		\do	156, 157, 163, 165, 176, 177, 186, 188, 191, 844, 1089
\DefaultWidthofCheckBox	888, 1087	\docslist	156, 165, 168, 170, 171, 176,

			Symbols
184, 187, 192, 1089			
document environment	486, 1089	max	941
\documentclass	5, 6, 168, 639, 1090	maxdepth	942
\doforrandN	664, 1090	upperbarlabel	942
\dolistcsloop	176, 1090	variable	941
\dolistloop	176, 1090	verticalbars	943
\dotfill	741, 810, 812, 813, 1090	ylabel	943
draft (changes)	983	yticgap	943
draft (probsln)	639	yticlabels	943
draftmark package	423	yticpoints	943
\draw	905, 907, 910, 963, 1090	\DTLbarchartlength	944, 1093
drm package	767	\DTLbarlabeloffset	944, 1093
\dropoints	617, 1091	\DTLbaroutlinecolor	945, 1093
\DTLabs	68, 1091	\DTLbaroutlinewidth	945, 947, 1093
\dtlabs	68, 1091	\DTLbarwidth	944, 946, 947, 1093
\DTLadd	66, 1091	\DTLbibfield	358, 1094
\dtladd	66, 1092	\DTLbibfieldlet	358, 1094
\DTLassign	196, 197, 205, 1092	\dtlbreak	151, 367, 1094
\DTLassignfirstmatch	196, 200, 201, 1092	\dtlcompare	111, 1094
\DTLbaratbegintikz	945, 1092	\DTLcustombibitem	359, 361, 365, 1094
\DTLbaratendtikz	945, 1093	\dtldefaultkey	77, 78, 1095
\DTLbarchart	940–943, 945, 947, 1093	\dtldisplayafterhead	147, 1095
axes	942	\DTLdisplaydb	78, 84, 137–139, 143, 145–147, 152, 213, 215, 1095
barlabel	942	\dtldisplayendtab	147, 1095
length	941	\DTLdisplaylongdb	138, 143, 145, 147, 324, 1095

		Symbols
caption	138	A N
contcaption	138	B O
foot	138, 148	C P
label	138	D Q
lastfoot	138	E R
omit	138	F S
shortcaption	138	G T
\dtldisplaystarttab	147, 1095	H U
\DTLdiv	68, 1095	I V
\dtldiv	68, 1096	J W
\DTLdoobarcolor	944, 1096	K X
\DTLdocurrentpiesegmentcolor	921, 1096	L Y
\DTLdopiesegmentcolor	921, 1096	M Z
\DTLendpt	945, 1096	
\DTLEverybarhook	945, 1097	
\dtlexpandnewvalue	688, 1097	
\DTLforeach	84, 149, 184, 185, 196, 304, 357, 367, 478, 677, 690, 693, 914, 921, 941, 1097	
\DTLforeach*	149, 152, 153, 156, 180, 185, 197, 215, 216, 270, 271, 275, 278, 296, 301, 325, 328, 329, 1097	
\DTLforeachbibentry	357–359, 367, 1097	
	\DTLformatbibentry 359, 361, 366, 367, 1098	
	\DTLformatthisbibentry 359, 367, 1098	
	\DTLgabs 68, 1098	
	\DTLgadd 66, 1098	
	\DTLgdiv 68, 1098	
	\DTLgmul 67, 1098	
	\DTLgneg 69, 1098	
	\DTLground 63, 1099	
	\DTLgsub 67, 1099	
	\dtlicompare 111, 1099	
	\DTLifbibfieldexists 358, 1099	
	\DTLifdbempty 363, 1099	
	\DTLiffirstrow 184, 185, 328, 921, 924, 1099	
	\DTLifinlist 161, 1100	
	\DTLiflastrow 185, 328, 1100	
	\DTLifnull 209, 215, 1100	
	\DTLifnullorempty 209, 215, 216, 272, 276, 296, 301, 1100	
	\dtlifnumeq 695, 1100	
	\dtllastloadeddb 83, 84, 1101	
	\dtlletterindexcompare 112, 1101	
	\DTLloadbb1 355, 360, 364, 1101	
	\DTLloaddb 73, 75–77, 79, 81, 82, 85, 87,	

			Symbols
113, 119–123, 139, 144, 205, 213, 271, 274, 294, 299, 300, 477, 918, 922, 946, 1101	groupgap	943	A N
autokeys	length	941	B O
headers	max	941	C P
keys	maxdepth	942	D Q
noheader	multibarlabels	942	E R
omitlines	uppermultibarlabels	942	F S
\DTLloaddbtx 83, 84, 98–101, 114, 126, 136, 141, 142, 145, 214, 673, 676, 689, 1101	variables	941	G T
\DTLloadrawdb 73, 75–77, 80, 85, 87, 1101	verticalbars	943	H U
autokeys	ylabel	943	I V
headers	yticgap	943	J W
keys	yticlabels	943	K X
noheader	yticpoints	943	L Y
omitlines	\DTLneg	69, 1103	M Z
\DTLmaketabspace 71, 1102	\dtlneg	69, 1103	
\DTLmidpt 945, 1102	\DTLnewdb	687, 1103	
\DTLmonthname 361, 1102	\DTLnewdbentry	688, 1104	
\DTLmul 67, 1102	\DTLnewrow	688, 1104	
\dtlmul 67, 1102	\DTLnumitemsinlist	161, 1104	
\DTLmultibarchart 941–943, 945, 1103	\DTLpieatbegintzikz	926, 1104	
axes 942	\DTLpieatendtzikz	926, 1104	
barlabel 942	\DTLpiechart	914, 1104	
	cutaway	916	
	cutawayoffset	916	
	cutawayratio	916	
	innerlabel	916	
	inneroffset	915	

			Symbols
innerratio	915	\DTLsort*	A N
outerlabel	916	\DTLstartpt	B O
outeroffset	915	\DTLsub	C P
outerratio	915	\dtlsub	D Q
radius	915	\dtlwordindexcompare	E R
rotateinner	917	\DTMdate	F S
rotateouter	917	\DTMnow	G T
start	914	\DTMsavedate	H U
variable	914, 917	\DTMsavetimestamp	I V
\DTLpiepercent	918, 1104	\DTMuse	J W
\DTLpievariable	917, 1105	\DTMusedate	K X
\DTLprotectedsaverawdb	671, 1105	\DTMusetime	L Y
\DTLradius	915, 1105	dvipdfm	M Z
\DTLrawmap	73, 76, 80, 1105	dvips	
\DTLround	63, 65, 1105	dynamiccontents environment	
\dtlround	62, 63, 65, 152, 153, 1106	762, 1109	
\DTLsaverawdb	671, 687, 690, 1106	E	
\DTLsetbarcolor	944, 946, 1106	\eappto	
\DTLsetdelimiter	71, 72, 1106	\EBC	
\DTLsetheader	84, 136, 142, 1106	\EBCi	
\DTLsetnumberchars	64, 1106	\ecvaddress	
\DTLsetpiesegmentcolor	917, 1107	\ecvafterpicture	
\DTLsetseparator	71, 72, 1107	\ecvAOne	
\DTLsettabseparator	71, 1107	\ecvATwo	
\DTLsort	112–114, 361, 365, 1107	\ecvbeforepicture	
\dtlsort	110, 1107	\ecvBOne	

			Symbols
\ecvBTwo	370, 1111	\edef 41–44, 447, 507, 511, 557, 561, 568, 1114	A N
\ecvCEF	369, 370, 1111	\EFC 315–317, 1115	B O
\ecvColSep	371, 1111	\EFCi 316, 1115	C P
\ecvCOne	370, 1111	eforms package 871	D Q
\ecvCTwo	370, 1111	email (isodoc) 265	E R
\ecvdateofbirth	345, 347, 350, 1111	\emailfrom 252, 274, 300, 377, 380, 1115	F S
\ecvemail	344, 347, 350, 1112	\emph 33–36, 38, 39, 144, 325, 328, 329, 1115	G T
\ecvfax	344, 1112	empty 169, 199, 209	H U
\ecvfootername	344, 1112	\empty 967, 1115	I V
\ecvgender	345, 347, 350, 1112	\encl 222, 223, 228, 246, 273, 286, 297, 1115	J W
\ecvitem	349–351, 353, 355, 365, 1112	\encllist 275, 300, 1115	K X
\ecvlanguage	368, 370, 1112	enclosures (isodoc) 264	L Y
\ecvlanguagefooter	369, 370, 1112	\end 31, 32, 268, 286, 297, 479, 865, 919, 924, 1116	M Z
\ecvlanguageheader	368, 370, 1113	\endfirsthead 323, 1116	
\ecvlastlanguage	369, 370, 1113	\endfoot 324, 325, 328, 329, 1116	
\ecvLeftColumnWidth	371, 1113	\endhead 323, 325, 328–330, 1116	
\ecvmothertongue	368, 370, 1113	\endinput 493, 822, 837, 870, 1116	
\ecvname	344, 347, 350, 1113	\endlastfoot 324, 325, 328, 329, 1116	
\ecvnationality	345, 347, 350, 1113	\endtime 396, 398, 1116	
\ecvpersonalinfo	346, 347, 350, 351, 354, 365, 1114	english (babel) 394	
\ecvpicture	345, 347, 350, 1114	\enspace 610, 612, 1117	
\ecvsection	349–351, 353, 355, 365, 1114	\enumerate 446	
\ecvspace	346, 1114		
\ecvtelephone	344, 347, 350, 1114		

Index

			Symbols
enumerate environment	437, 443, 448, 588, 635, 636, 645, 1117	personal envelope	A N
envbig package	279	rotate envelope	B O
environ package	867, 868	rotate envelopes	C P
envlab package	279, 284, 286, 291, 298, 304	EOL 5, 6, 76, 85, 168, 414, 642, 646, 1051, 1052, 1056, 1057	D Q
avery5160label	283	\epreto 52, 1117	E R
avery5161label	283	\equal 151, 278, 504, 1117	F S
avery5162label	283	etextools package 165, 166	G T
avery5163biglabel	284	etoolbox package 30, 43, 48, 156, 166, 172, 209, 1130	H U
avery5163label	283	europecv environment 342, 346, 351, 363, 1117	I V
avery5164biglabel	284	europecv class 335, 340, 368	J W
avery5164label	283	booktabs 371	K X
avery5262label	283	helvetica 340	L Y
booklet envelope	282	narrow 340	M Z
business envelope	282	nologo 340	
c5 envelope	282	notitle 340	
c65 envelope	282	totpages 343	
c6 envelope	282		
capaddress	291, 292	\evensidemargin 816, 817, 1117	
d1 envelope	282	\everypar 450, 456, 1117	
executive envelope	282	Evince 878	
herma4625label	283	exam class 606, 608, 644, 646, 679	
no capaddress	291, 292	addpoints 608, 609, 614, 618	
no rotate envelope	286	answers 609, 619, 625	
no rotate envelopes	281	Excel spreadsheet (.xls) 83, 87, 100,	

		Symbols
123, 145, 152	.dbtex	A N
executive envelope (<code>envelope</code>)	282	B O
\expandafter	282	C P
159, 161, 448, 476, 480,	.jdr	D Q
500, 501, 507, 508, 512, 515–517,	.ods	E R
845, 1118	.odt	F S
\expandonce	53, 1118	G T
exsheets package	606, 626, 629, 638	H U
extended character	72, 291, 1053, 1056	I V
extended Latin character	1053, 1056	J W
F		K X
fax (<code>isodoc</code>)	265	L Y
\fbox	527, 547, 551, 559, 565, 610, 712,	M Z
1118	314, 317, 1118	
\Fee	818	
\ffareax	484	
\fi		
field		
see column		
figure environment	31, 220, 1118	
file formats		
.ajr	789, 793, 802	
.aux	698	
.csv	21, 69–73, 75–81, 83, 85, 87, 98, 112, 113, 115, 117, 123, 127, 144, 209, 313, 320	
flowchart package	898	
flowfram package	709, 758, 765, 767, 777, 802, 817, 1053	
pages=absolute	761	
flowframtk	709, 710, 772, 774, 793, 802,	
1053		
flushright environment	736, 1119	
foldmark (<code>leaflet</code>)	733	

			Symbols
foldmarks (KOMA option)	235, 243	\form@layout@fillin	834, 836
fontenc package	72, 314, 320, 342, 1057	Foxit	878
T1	314	\fp package	61, 700
\fontfamily	431, 1119	\FPrandom	700, 1121
\fontseries	432, 1119	\FPseed	700, 1122
fontspec package	72, 1057	\frac	668, 669, 1122
footer (isodoc)	265	\frame	755, 1122
\footnote	51, 1120	frame environment	570, 573, 575, 577,
footsepline (KOMA option)	242	1122	
\for@block	851	\framebox	717, 1122
forcedate (isodoc)	263	\framebreak	765, 1122
\forcsvlist	157, 160, 193, 1120	framed package	810
\foreach	167, 170, 553, 558, 559, 565,	\framesubtitle	575, 1123
954, 1120		\frametitle	573, 577, 1123
\foreachdataset	659, 660, 1120	fromaddress (KOMA variable)	236
\foreachproblem	657, 670, 706, 1120	fromalign (KOMA option)	237
\foreachsolution	658, 670, 706, 1121	fromemail (KOMA option)	238
foreign (isodoc)	262	fromemail (KOMA variable)	236
\foreignfalse	262	fromfax (KOMA option)	238
\foreigntrue	262	fromfax (KOMA variable)	236
\forlistcsloop	177, 1121	fromlogo (KOMA option)	238
\forlistloop	166, 177, 192, 1121	fromlogo (KOMA variable)	236, 244
Form environment	873, 1121	frommobilephone (KOMA option)	237
\form@block	856	frommobilephone (KOMA variable)	236
\form@fillin	825, 836	fromname (KOMA variable)	236
\form@layout@checkbox	834, 836	fromphone (KOMA option)	237

		Symbols
fromphone (KOMA variable)	236	A N
fromurl (KOMA option)	238	B O
fromurl (KOMA variable)	236	C P
frontside (leaflet)	733	D Q
fullmemo (newlfm)	374	E R
G		F S
\ganttbar	955, 957, 1123	G T
ganttchart environment	950, 953, 1123	H U
time slot format	951	I V
\ganttgroup	955, 957, 1123	J W
\ganttlinkedbar	956, 1123	K X
\ganttlinkedgroup	956, 1124	L Y
\ganttlinkedmilestone	956, 1124	M Z
\ganttmilestone	955, 957, 1124	
\ganttnewline	951, 953, 956, 1124	
\ganttset	950, 1124	
\ganttttitle	953, 1124	
\ganttttitlecalendar	954, 957, 1124	
\ganttttitlecalendar*	954, 1125	
\ganttttitlelist	954, 1125	
\gappto	48, 1125	
\gdef	40, 41, 43, 44, 183, 849, 867, 869, 1125	
\gDTLforeachbibentry	357–359, 361, 365, 366, 1125	
		H
	\gender	858
	geometry (ifsym)	831, 832
	geometry package	221, 429, 720, 728, 767, 777
	\getflowbounds	818, 1125
	\getflowevenbounds	818, 1126
	gitinfo2 package	991
	\global	34, 63, 150, 183, 556, 560, 566, 665, 666, 675, 704, 705, 849, 1126
	Google Chrome	878
	GOW	992, 1053
	\gpreto	52, 1126
	\gradetable	618, 1126
	graphics package	244, 289, 340, 720, 725, 728, 896
	gray (databar)	940
	gray (datapie)	913
	\greetto	250, 276, 302, 1126
	\guest	396, 398, 1126
	GUI	86, 92, 93, 113, 127, 671, 709, 772, 1054
	\half	608, 609, 614, 1126
	\headline	380, 381, 1127

		Symbols
headsepline (KOMA option)	241	A N
helvetica (europecv)	340	B O
herma4625label (envlab)	283	C P
\hfill	289, 295, 1127	D Q
hiddenitems environment	391, 1127	E R
hiddensubitems environment	391, 1127	F S
hiddentext environment	392, 1127	G T
\hideanswers	640, 1127	H U
\hoffset	815–817, 1127	I V
horizontal (dbar)	940	J W
\hrulefill	610, 612, 810, 813, 824, 836, 869, 1128	K X
\hspace	289, 295, 815, 817, 1128	L Y
\Huge	738, 1128	M Z
hyperref package	107, 871, 874	
pdfauthor	107	
hypersetup	107, 1128	
I		
iban (isodoc)	307	
ICO	115, 1054	
\ifbool	641, 675, 1128	
\ifboolexpr	211, 299, 1128	
\ifcase	484, 696, 1129	
\ifcsbool	212, 216, 299	
\ifcsdef	46, 549, 564, 829, 831, 836, 839,	
	841, 1129	
	\ifcsundef	
	47, 1129	
	\ifdate	
	522, 530, 533, 534, 537, 539, 548, 552, 555, 559, 561, 565, 567, 1129	
	\ifdef	
	46, 494, 497, 666, 1130	
	\ifdefempty	
	209, 215, 507, 511, 1130	
	\ifdefstring	
	210, 299, 1130	
	\ifinlist	
	175, 1130	
	\ifinlistcs	
	176, 1130	
	\ifnum	
	179, 447, 513, 555, 559, 565, 695, 1131	
	\ifnumless	
	366, 367, 1131	
	\ifodd	
	815, 817, 818, 1131	
	\ifshowanswers	
	641, 648, 1131	
	\ifstempty	
	489, 495, 1131	
	\ifstreq	
	503, 514, 1132	
	\ifsym package	
	831, 832	
	geometry	
	831, 832	
	\ifthen package	
	151, 278, 641, 673	
	\ifthenelse	
	151, 366, 504, 641, 675, 1132	
	\ifthispageodd	
	816, 1132	
	\ifundef	
	47, 663, 704, 705, 1132	
	\ignorespaces	
	646, 1132	
	\ignorespacesafterend	
	646, 1133	

Index

		Symbols
\iitem	308, 310, 1133	A N
\include	976, 1133	B O
\includegraphics	244, 247, 284, 385, 424, 432, 573, 589, 592, 721, 725, 729, 738, 751, 757, 764, 768, 793, 796, 1133	C P
\includequestions	637, 1133	D Q
\incorrectitem	646, 1133	E R
\incorrectitemformat	646, 1134	F S
inparaenum environment	635, 1134	G T
\input	83, 84, 656, 976, 1134	H U
inputenc package	72, 73, 76, 314, 1050, 1057	I V
utf8	72	J W
\InputIfFileExists	421, 687, 1134	K X
\inst	572, 592, 1134	L Y
\institute	571, 573, 592, 1135	M Z
internal command	12, 43, 56, 163, 447, 457, 1054	
\invoice	306, 308, 310, 1135	
invoice environment	313, 1135	
invoice (KOMA variable)	236	
invoice package	305, 313, 319, 320	
isodoc class	220, 260, 268, 278, 305, 306, 321	
accept	307	
acceptaccount	307	
acceptaddress	307	
acceptcents	307	
acceptdescription	307	
accepteuros	307	
acceptpreference	308	
accountname	307	
accountno	306	
areacode	262	
autograph	264, 268	
bic	307	
cellphone	265	
city	261	
cityzip	262	
closing	264	
company	261	
copyto	264	
country	262	
countrycode	261	
currency	306	
date	263	
email	265	
enclosures	264	
fax	265	
footer	265	
forcedate	263	

			Symbols
foreign	262	670, 677, 678, 690, 694, 697, 742,	A N
iban	307	770, 806, 1135	B O
language	261, 263	itemize environment	C P
logoaddress	261	items environment	D Q
opening	264	\itotal	E R
ourref	263	\itshape	F S
phone	265		G T
phoneprefix	265	J	H U
return	262	Java	I V
returnaddress	262	\labels package	J W
routingno	307	\jobname	K X
signature	264		L Y
street	261	K	M Z
subject	264	key	
term	306	see column label	
to	262	key=value list	2, 1050, 1054
vatno	307	keyval package	640
website	265	keyword anchor	992
who	261	KOMA-Script class options	
yourletter	263	parskip	230
yourref	263	KOMA-Script options	
zip	261	addrfield	238
\itable	308, 310, 312, 321, 333, 1135	backaddress	238
\item	337, 359, 391, 393, 403, 404, 441, 575, 579, 580, 583, 586, 588, 589, 593, 636, 645, 646, 655, 658–661,	firsthead	237
		foldmarks	235, 243
		footsepline	242

			Symbols
fromalign	237	fromphone	A N
fromemail	238	fromurl	B O
fromfax	238	invoice	C P
fromlogo	238	location	D Q
frommobilephone	237	myref	E R
fromphone	237	nextfoot	F S
fromurl	238	nexthead	G T
headsepline	241	signature	H U
locfield	239	subject	I V
numbers=endperiod	450	title	J W
numericaldate	239	yourref	K X
pagenumber	242	\KOMAoption	L Y
priority	239	\KOMAoptions	M Z
refline	240	kpsewhich	
subject	234, 241		
KOMA-Script variables			
customer	236	\label{16, 19, 56, 144, 146, 153, 213–215, 277, 323, 422, 438, 442, 453, 458, 478, 636, 1136}	
date	236	\label{counter} 438	
firstfoot	236	\label{enumi} 438, 440, 645, 1137	
firsthead	236	\label{enumii} 440, 645, 1137	
fromaddress	236	\label{enumiii} 439, 440, 1137	
fromemail	236	\label{enumiv} 440, 1137	
fromfax	236	\LabelHeight 288, 289, 295, 1137	
fromlogo	236, 244	labels package 708	
frommobilephone	236		
fromname	236		

			Symbols
LabelsAligned (currvita)	335	\leftmargin	A N
\labelsep	445, 1137	\LenToUnit	B O
\LabelWidth	288, 289, 295, 1137	\let	C P
\labelwidth	445, 1138	547, 551, 559, 565, 665, 666, 704,	D Q
language (isodoc)	261, 263	705, 839, 849, 1139	E R
\Large	33, 676, 678, 689, 726, 729, 768, 793, 1138	\letcs	F S
Large environment	32, 1138	\letter	G T
\large	721, 725, 729, 1138	letter environment	H U
latexmk	109	221, 227, 245, 260, 270, 1074, 1078, 1115, 1140, 1156, 1172	I V
\LayoutCheckField	879, 892, 893, 1138	letter class	J W
\LayoutChoiceField	879, 1138	\lim	K X
\LayoutTextField	879, 1138	\line	L Y
leaders	813	\lineskip	M Z
leaflet class	709, 730, 733, 734	\linewidth	
backside	733	330, 768, 1141	
bothsides	733	\lipsum	
combine	733	426, 434, 451, 458, 764, 1141	
foldmark	733	lipsum package	
frontside	733	425, 433, 451, 763	
nocombine	733	\list	
nofoldmark	733	448	
notumble	732	list environment	
notwopart	733	443, 1141	
tumble	732	list or summary of changes file (.soc)	
twopart	733, 743	983	
		\listadd	
		172, 174, 1141	
		\listcsadd	
		173, 174, 1141	
		\listcseadd	
		174, 175, 1142	
		\listcsgadd	
		174, 1142	
		\listcsxadd	
		175, 1142	

			Symbols
\listeadd	173, 174, 1142	1145	A N
\listgadd	174, 1142	longtable package	B O
listings package	673	\loop	C P
\listofchanges	983, 1142	\listinputlisting	D Q
style	983	lstlisting environment	E R
\listofdecisions	407, 1143	M	F S
\listoffigures	983, 1143	make	G T
\listoftables	407, 1143	\makeatletter	H U
\listxadd	174, 1143	894, 1054, 1145	I V
\llap	827, 1143	\makeatother	J W
lmodern package	728	43, 163, 170, 447, 457, 894,	K X
\loadallproblems	656, 665, 1052, 1143	1054, 1146	L Y
\LoadClass	820, 835, 868, 1143	\makebox	M Z
\loadexceptproblems	656, 1144	610, 612, 717, 721, 726, 729,	
\loadrandomexcept	657, 1144	814–817, 824, 836, 869, 1146	
\loadrandomproblems	657, 660, 670, 1144	\MakeButtonField	
\loadselectedproblems	656, 1144	887, 1146	
\location	226, 271, 285, 295, 396, 398, 1144	\MakeCheckField	
location (KOMA variable)	236, 239	886, 891, 1146	
locfield (KOMA option)	239	\MakeChoiceField	
logoaddress (isodoc)	261	887, 1146	
\long	43, 869, 1145	makeindex	
longtable environment	138, 144, 145, 147, 148, 321, 333, 351, 360, 366,	979	
		\makelabels	
		279, 281, 285, 295, 1146	
		\MakeRadioField	
		886, 1147	
		makeshape package	
		898	
		\MakeTextField	
		886, 1147	
		\maketitle	
		336, 390, 397, 571, 1147	
		\MakeUppercase	
		304, 1147	
		ManyBibs (currvita)	
		336	
		\marginpar	
		459, 693, 1147	

		Symbols
markup (changes)	983	A N
<code>math=pgfmath</code> (datatool)	62	B O
<code>\mbox</code> 552, 556, 558, 560–562, 566–568, 1147		C P
mdframed package	810	D Q
<code>\medskip</code>	936, 1147	E R
meetingmins class	387, 388, 394, 402	F S
agenda	388, 389, 392	G T
chair	388, 389, 392, 394	H U
notes	388, 392, 394	I V
<code>memonofrom</code> (newlfm)	375	J W
<code>memonore</code> (newlfm)	375	K X
<code>memonoto</code> (newlfm)	375	L Y
<code>\midrule</code>	148, 330, 331, 1148	M Z
minipage environment	288, 1141, 1148 , 1158	
minitoc package	399	
Minutes environment	394, 409, 1148	
minutes package	387, 394, 402, 409	
Secret	402	
<code>\minutesdate</code>	396, 398, 1148	
<code>\minutetaker</code>	395, 398, 1148	
<code>\missing</code>	397, 398, 1149	
<code>\missingExcused</code>	397, 1149	
<code>\missingNoExcuse</code>	397, 1149	
<code>\mlabel</code>	298, 303, 1149	
<code>\moderation</code>	395, 398, 1149	
<code>\month</code>	470, 482, 498, 501, 1149	
multi-line cell	76, 85	
multibib package	336	
<code>\multicolumn</code>	153, 215, 325, 328, 329, 331, 1149	
multiple choice	622	
<code>\multiply</code>	60, 1149	
<code>\multiput</code>	720, 1150	
<code>mwe</code> package	247	
<code>\myref</code> (KOMA variable)	236	
N		
<code>\n</code>	1052	
<code>\name</code>	224, 271, 285, 295, 1150	
<code>\namefrom</code>	251, 274, 300, 377, 381, 1150	
<code>\nameto</code>	250, 276, 301, 377, 1150	
<code>narrow</code> (europecv)	340	
<code>natbib</code> package	353	
<code>\NeedsTeXFormat</code>	492, 820, 835, 868, 1150	
<code>\newbool</code>	673, 1150	
<code>\newboolean</code>	673, 1150	
<code>\newcommand</code>	24, 31, 36, 37, 40, 696, 867, 1151	
<code>\newcount</code>	55, 179, 470, 478, 488,	

		Symbols
495, 529, 532, 533, 536, 537, 551, 556, 559, 569, 662, 848, 853, 856, 1151	datecenter dateleft dateno dateright dateyes fullmemo memonofrom memonore memonoto noaddrfrom orderdatefromto orderfromdateto orderfromtodate pressrelease printallfrom printallto sigcenter sigleft sigright stdletter stdletterfrom stdmemo useenvlab	256 255 256 255 256 374 375 375 375 255 256 256 380 255 255 256 256 256 256 254, 374 254 374 298 254, 274, 300, 375, 1152 344, 564, 1152 736, 739, 1152
\newcounter 56, 360, 364, 443, 444, 451, 453, 1151		A N
\newdynamicframe 760, 768, 1151		B O
\newenvironment 864, 865, 869, 1151		C P
\newflowframe 760, 764, 768, 1151		D Q
\newglossaryentry 44, 1152		E R
\newlength 444, 445, 818, 1152		F S
newlfm environment 249, 260, 270, 277, 374, 1152		G T
newlfm class 220, 249, 256, 268, 298, 374, 378		H U
addrfromemail 255		I V
addrfromfax 255		J W
addrfromleft 255		K X
addrfromphone 255		L Y
addrfromright 255		M Z
addrtoemail 255		
addrtofax 255		
addrtoleft 255		
addrtophone 255		
addrtright 255		
busletter 254	\newline	
busletternofrom 254	\newpage	

			Symbols
\newproblem	653, 1152	\noprintanswers	609, 1154
\newproblem*	654, 1152	\normalsize	738, 1154
\newstaticframe	760, 764, 1153	norotateenvelope (envlab)	286
\newwatermark	429, 433, 1153	norotateenvelopes (envlab)	281
nextfoot (KOMA variable)	236	norotateinner (datapie)	913
nexthead (KOMA variable)	236	norotateouter (datapie)	914
\nextmeeting	392, 393, 1153	notes (meetingmins)	388, 392, 394
\noaddpoints	608, 1153	notitle (europecv)	340
noaddrfrom (newlfm)	255	notumble (leaflet)	732
noanswers (probsoln)	638, 640	notwopart (leaflet)	733
\NoBgThisPage	424, 1153	nousedefaultargs (probsoln)	639
\nobibliography	353, 354, 1153	null	112, 199, 209
nocapaddress (envlab)	291, 292	\number	59, 478, 482, 483, 489, 496, 509,
\nocite	339, 353, 354, 356, 360, 364, 1154	556, 557, 560, 561, 566–568, 690, 1155	639
nocombine (leaflet)	733	numbers=endperiod (KOMA option)	450
NoDate (currvita)	336	numericaldate (KOMA option)	239
\node	526, 899, 910, 1154	\numexpr	57, 555, 557, 560, 561, 566, 568, 1155
\nodepart	542, 543, 545, 546, 548, 553, 556, 558, 560–562, 566–568, 1154	\numparts	617, 1155
\noexpand	51, 1154	\numpoints	618, 1155
nofancy (svninfo)	999	\numquestions	617, 1155
nofoldmark (leaflet)	733	\numsubparts	618, 1155
\noindent	644, 647, 814, 1154	\numsubsubparts	618, 1155
nologo (europecv)	340		

O	\oval	713, 737, 1157	Symbols
\oddsidemargin	815–817, 1155		A N
Okular	878		B O
\onecolumn	758, 1156		C P
oneparcheckboxes environment	623, 1156		D Q
oneparchoices environment	623, 1156		E R
onlyproblem environment	641, 643, 650, 658, 1156		F S
onlysolution environment	641, 644, 645, 650, 658, 1156		G T
Open Document Format (.odt)	977		H U
Open Document Spreadsheet (.ods)	83, 91, 126		I V
openbib (currvita)	336		J W
\opening	221, 223, 226, 228, 233, 245, 272, 285, 296, 1156		K X
opening (isodoc)	264		L Y
\opinion	403, 1156		M Z
Opinions environment	403, 1156		
\or	484		
\ord	485, 495		
orderdatefromto (newlfm)	256		
orderfromdateto (newlfm)	256		
orderfromtodate (newlfm)	256		
OT1	314		
ourref (isodoc)	263		
	\oval	713, 737, 1157	
	\p@{counter}	438	
	\p@{enumi}	440, 1157	
	\p@{enumii}	439, 1157	
	\p@{enumiii}	439, 1157	
	\p@{enumiv}	438, 1157	
	page style		
	empty	767	
	pagenumber (KOMA option)	242	
	pages=absolute (flowfram)	761	
	\pagestyle	720, 768, 780, 1157	
	\paperheight	735, 737, 1157	
	\paperwidth	735, 737, 815–817, 1158	
	\par	4, 644, 645, 677, 690, 768, 814, 837, 1158	
	\paragraph	454, 1158	
	paralist package	607, 635	
	\parbox	553, 556, 558, 560–562, 566–568, 610, 892, 946, 947, 1141, 1148, 1158	
	\parindent	55, 288, 1159	
	\parsemdydate	476	
	\parshape	760, 1159	
	parskip (KOMA)	230	

			Symbols
\part	613, 616, 732, 1159		A N
\participant	395, 398, 1159		B O
parts environment	613, 1159		C P
\PassOptionsToClass	820, 835, 868, 1159		D Q
\path	526, 547, 551, 555, 558–560, 562, 565, 566, 568, 747, 833, 899, 951, 963, 1160	\pgfcalendar package 521, 524, 525, 527, 548, 549, 552, 555–557, 559, 565, 1160	E R
color	964	\pgfcalendar package 481, 521, 950	F S
dashdotted	964	\pgfcalendarbeginiso 522, 1160	G T
dashed	964	\pgfcalendarbeginjulian 522, 1160	H U
dotted	964	\pgfcalendarcurrentraday 522, 555, 1161	I V
mark	963	\pgfcalendarcurrentjulian 521, 557, 561, 567, 1161	J W
mark repeat	964	\pgfcalendarcurrentmonth 522, 1161	K X
mark size	964	\pgfcalendarcurrentweekday 521, 527, 548, 552, 555, 557, 559, 561, 565, 567, 1161	L Y
no markers	964	\pgfcalendarcurrenyear 522, 1161	M Z
thick	964	\pgfcalendardatetojulian 470, 478–480, 487, 488, 490, 494, 496, 497, 1161	
thin	964	\pgfcalendarendiso 522, 1162	
pdfauthor (hyperref)	107	\pgfcalendarendjulian 522, 1162	
\pdfcreationdate	499, 501, 508, 1020, 1023, 1160	\pgfcalendarifdate 472, 522, 1162	
\pdffilemoddate	502, 516, 517, 1160		
pdflatex	13		
\pdfnow	516		
\pdfnowtime	509, 515		
pdftk	106, 423		

		Symbols
\pgfcalendarjuliantodate	471, 488, 496, 555, 558, 560, 562, 566, 568, 1162	\pgfmathrandominteger 702, 705, 1164
\pgfcalendarjuliantoweekday	472, 487, 488, 490, 494, 496, 497, 1162	\pgfmathrandomitem 703, 1165
\pgfcalendarmonthname	481, 489, 496, 565, 1163	\pgfmathresult 701
\pgfcalendarmonthshortname	481, 1163	\pgfmathsetseed 702, 1165
\pgfcalendarprefix	522, 1163	pgfornament package 748
\pgfcalendarshorthand	523, 547, 551, 559, 565, 1163	pgfplots package 960
\pgfcalendarsuggestedname	523, 549, 1163	\pgfplotsset 960, 967, 969, 1165
\pgfcalendarweekdayname	481, 489, 496, 1164	axis lines* 971
\pgfcalendarweekdayshortname	481, 558, 559, 565, 1164	compat 961
pgffor package	166, 556	height 961
pgfgantt package	948, 1123 , 1124	major tick length 962
\pgfkeys	969, 1164	minor tick length 962
pgfkeys package	468	scale only axis 961
pgfmath package	61, 667, 700, 960, 969	scaled ticks 971
\pgfmathdeclarerandomlist	703, 1164	tick align 962
\pgfmathparse	701, 1164	title 961, 969
\pgfmathprintnumber	969, 1164	width 961
		x tick label as interval 967
		xlabel 961, 969
		xtick 967
		xticklabel 967
		xticklabels 967
		y tick label as interval 967
		ylabel 961, 969
		ytick 967
		ztick 967

Index

			Symbols
phone (isodoc)	265	Postscript environment	A N
\phonefrom	252, 274, 300, 377, 381, 1165	\postscript	B O
phoneprefix (isodoc)	265	\pounds	C P
picontoptext (xwatermark)	428	\psitem	D Q
picture environment	708–710, 719, 720, 722, 724, 734, 746, 1165	\PRaddress	E R
\pie	928, 931, 1166	\PRcompany	F S
after number	930	\PRdepartment	G T
auto	930	\PRmail	H U
before number	930	pressrelease environment	I V
color	929	pressrelease (newlfm)	J W
explode	929	pressrelease class	K X
pos	929	symbols	L Y
radius	929	\preto	M Z
rotate	929	51, 454, 1167	
scale font	930	\PRfax	
sum	929	\PRheadline	
text	930	\PRhours	
piechart package	911	384, 386, 1168	
piechartmp package	911	primitive	
pifont package	213, 767, 810, 814	27, 43, 46, 52, 55, 57, 58, 476, 493, 499, 501, 760, 840, 1023, 1055	
\points	634, 1166	\PrintAddress	
\pointsdroppedatright	617, 1166	288, 289, 295, 1168	
\pointsinmargin	616, 1166	printallfrom (newlfm)	
\pointsinrightmargin	617, 1166	255	
\pointsum	634, 1166	printallto (newlfm)	
		255	
		\printanswers	
		609, 647, 1168	
		\PrintBigLabel	
		288, 289, 295, 1168	
		\printdate	
		488, 498	
		\PrintReturnAddress	
		288, 289, 295,	

		Symbols
1168		
\printsolutions	632, 1168	A N
printwatermark (xwatermark)	428	B O
priority (KOMA option)	239	C P
\priormins	392, 1169	D Q
\PRlocation	385, 1169	E R
\PRlogo	384, 1169	F S
\PRmobile	386, 1169	G T
\Pro	405, 1169	H U
\pro	405, 1169	I V
probsln package	607, 638, 644, 646, 671, 679, 700, 1052, 1144	J W
answers	638, 640, 658, 659	K X
draft	639	L Y
final	639	M Z
noanswers	638, 640	
nousedefaultargs	639	
usedefaultargs	639, 657	
\ProbSolnFragileExt	640, 1169	
\ProbSolnFragileFile	640, 1170	
\ProcessOptions	821, 835, 868, 1170	
\project	859	
\ProjectTitle	314, 317, 1170	
proof environment	581, 1170	
\protect	698, 1170	
\protected@csedef	45, 1170	
\protected@csxdef	45, 1170	
\protected@edef	43, 45, 1171	
\protected@write	698, 1171	
\protected@xdef	43, 45, 1171	
\providecommand	36, 1171	
\ProvidesClass	820, 835, 868, 1171	
\ProvidesPackage	313, 492, 1171	
\PRphone	386, 1172	
\PRsubheadline	384, 1172	
\PRurl	386, 1172	
\ps	222, 223, 228, 246, 273, 286, 297, 1172	
ps2pdf	750	
\psbarcode	751, 754–756, 1172	
\psitem	253, 275, 300, 1172	
\PSNrandom	662, 700, 1172	
\PSNrandseed	661, 1172	
\pspicture environment	751, 1173	
pst-bar package	932	
pst-barcode package	748	
pst-gantt package	948	
pst-pdf package	751	
pstricks package	746, 748, 751, 896, 948	
\PushButton	874, 1173	
\put	711, 716, 718, 719, 721, 724, 729, 737, 747, 1173	

Q	Symbols
\qbezier	A N
719, 1173	
\qqquad	B O
837, 877, 923, 1173	
\quad	C P
1173	
\question	D Q
609, 612, 614–616, 621, 624, 648, 1174	
question environment	E R
627, 629, 630, 633, 637, 1173	
questions environment	F S
609, 619, 1174	
R	
\r	G T
1052	
\raggedright	H U
330, 736, 739, 768, 793, 806, 947, 1174	
\random	I V
662, 1174	
rcs package	J W
res-multi package	K X
resinfo package	L Y
\re	M Z
375, 377, 1174	
\ref	
16, 19, 56, 138, 416, 422, 438, 439, 443, 453, 478, 636, 1174	
refline (KOMA option)	1177
\refstepcounter	636, 1177
56, 360, 364, 451, 1175	
\regarding	438, 636, 1177
253, 274, 300, 375, 1175	
\relax	447, 1177
\release	289, 295, 737, 1177
380, 1175	
\renewcommand	rotateenvelope (envlab)
24, 36, 183, 867, 889, 935, 945, 1175	
\renewenvironment	286
446, 647, 1175	
\replaced	982, 987, 1175
id	
982	
remark	
982	
\RequirePackage	982
313, 493, 819, 823, 832, 836, 869, 1175	
\Reset	874, 1176
\reset@block	
852	
\reset@element	
850	
\resizebox	850
737, 833, 1058, 1176	
\RestartCensoring	
415, 1176	
\result	
405, 1176	
return (isodoc)	
262	
\returnaddress	
284, 1176	
returnaddress (isodoc)	
262	
\rightarrowarrow	
669, 1176	
\rlap	
815–817, 825, 826, 831, 836, 869, 1176	
\rmfamily	1177
\Roman	
636, 1177	
\roman	
438, 636, 1177	
\romannumeral	
447, 1177	
\rotatebox	
289, 295, 737, 1177	
rotateenvelope (envlab)	

			Symbols
rotateenvelopes (envlab)	281	\setbeamercovered	583, 1179
rotateinner (datapie)	913	\SetBigLabel	283, 285, 294, 1179
rotateouter (datapie)	914	\setbool	673, 1179
routingno (isodoc)	307	\setboolean	375, 673, 1179
row	70	\setcommittee	389, 1179
index (<row-idx>)	70	\setcounter	768, 780, 947, 1179
rtsched package	948	\setdate	389, 1180
\rule	288, 289, 295, 921, 924, 1177	\setdynamiccontents	762, 793, 1180
		\SetEnvelope	281, 1180
S		\setkeys	640, 1180
\sb	1177	\setkomavar	232, 234, 245, 246, 1180
scope environment	533, 909, 910, 1178	\setkomavar*	232, 245, 1180
scrartcl class	777	\SetLabel	282, 283, 1180
scrbook class	5	\setlength	444, 445, 712, 716, 718, 720,
scrlttr2 class	219, 227, 268	728, 943, 947, 1181	
\scshape	729, 737, 768, 793, 1178	\setmargins	732, 1181
Secret environment	402, 1178	\setmembers	389, 1181
\secret	402, 1178	\setpresent	389, 1181
Secret (minutes)	402	\setsocextension	983, 1181
\sectfont	734, 737, 1178	\setstaticcontents	762, 764, 768, 1181
\section	46, 351, 361, 380, 381, 388, 393, 401, 439, 441, 454, 458, 579, 593, 661, 670, 678, 739, 767, 770, 806, 870, 1178	\setupdocument	260, 263, 264, 266, 306, 309, 1181
\selectfont	1178	\SetupExSheets	626, 631–633, 635, 1182
\set@element	850	\sf	935
		\sffamily	935, 1182
		shapepar package	760

		Symbols
shell escape	107–109, 748, 750, 958, 963, 1055	A N
\shortstack	715, 721, 1182	B O
\show	25, 28, 446, 455, 1182	C P
\showanswers	640, 647, 658, 659, 670, 706, 1182	D Q
showanswers boolean flag	641, 643, 645	E R
showdow (datetime2)	465	F S
showpagecenter (xwatermark)	428	G T
sigcenter (newlfm)	256	H U
sigleft (newlfm)	256	I V
\signature	224, 271, 285, 295, 409, 1182	J W
signature (KOMA variable)	236	K X
signature (isodoc)	264	L Y
sigright (newlfm)	256	M Z
\sin	622, 652, 662, 664, 668, 705, 1182	
\small	416, 553, 561, 567, 1183	
\smallskip	936, 1183	
solution environment	625, 630, 637, 644–646, 652, 1183	
\solutionname	644, 647, 1183	
solutionorbox environment	620, 625, 1183	
solutionordottedlines environment	621, 1184	
solutionorlines environment	620, 1184	
\sp	1184	
sp (scaled point)	55	
\space	295, 451, 475, 489, 496, 648, 839, 1184	
spaces	22, 162	
special character	25, 70, 77, 78, 86, 87, 651, 656, 1055	
SQL	21, 69, 93, 104, 106, 112–115, 123, 126, 127, 142, 144, 150, 151, 199, 200, 206, 208–210, 215, 312, 320, 333, 686, 926, 1052, 1056	
\Square	735, 822, 832, 1185	
standalone class	755	
\startlabels	298, 302, 1185	
\starttime	396, 398, 1185	
staticcontents environment	762, 1185	
stdletter (newlfm)	254, 374	
stdletterfrom (newlfm)	254	
stdmemo (newlfm)	374	
\STExpenses	317, 1185	
\StopCensoring	415, 417, 1185	
street (isodoc)	261	
\string	839, 840, 1185	
subitems environment	391, 1186	
subject (KOMA option)	234, 241	
subject (KOMA variable)	236	
subject (isodoc)	264	

			Symbols
\Submit	874, 1186	\svnInfoMaxToday	A N
\ subparagraph	454, 1186	\svnInfoMinRevision	B O
\subpart	614, 616, 1186	\svnInfoMonth	C P
subparts environment	613, 1186	\svnInfoOwner	D Q
\subsection	388, 454, 458, 579, 593, 1186	\svnInfoRevision	E R
\subsubsection	614, 616, 1187	\svnInfoTime	F S
subsubparts environment	614, 1187	\svnInfoYear	G T
\subsubsection	454, 1187	\svnKeyword	H U
\subtitle	395, 398, 571, 572, 592, 1187	symbols (pressrelease)	I V
\subtopic	401, 402, 1187	T	J W
Sumatra	878	T1 (fontenc)	K X
svn package	991	tab character 	L Y
svn-multi package	991	table environment	M Z
svn-prov package	991	137–139, 147, 220, 1190	
svn:keywords	999	\tableofcontents	
\svnid	1001, 1187	399, 401, 577, 983, 1190	
\svnInfo	995, 997, 999, 1001, 1187	tabular environment	
svninfo package	991, 994	137, 138, 145, 150, 171, 178, 183, 197, 220, 321, 329, 357, 366, 410, 414, 619, 647, 715,	
nofancy	999	810, 848, 855, 890, 891, 893, 922, 1061, 1074, 1148, 1190, 1197	
\svnInfoDate	1000, 1188	tabularx package	
\svnInfoDay	1001, 1188	\task	
\svnInfoFile	1000, 1188	\task*	
\svnInfoHeadURL	1002, 1188	tdclock package	
\svnInfoLongDate	1001, 1188		
\svnInfoMaxRevision	1002, 1188		

Index

			Symbols
\telephone	226, 271, 285, 295, 1191	\texttt	35, 39, 146, 153, 1193
term (isodoc)	306	\textwidth	58, 361, 610–612, 764, 768,
\TeX	518, 1191	1193	
tex		TeXworks	878
-jobname	1136	\the	58, 1193
-no-shell-escape	108	\the<counter>	438
-shell-escape	108	thebibliography environment	1193
-shell-restricted	108	\theenumi	438, 440, 647, 1193
tex4ht	977	\theenumii	440, 1194
texdef	28, 446, 455, 1056	\theenumiii	438, 440, 1194
texdoc	305	\theenumiv	438, 440, 1194
\text	513, 1191	themes	590
TextAligned (currvita)	335	theorem environment	581, 1194
\textasciicircum	74, 1191	\thequestion	610, 1194
\textasciitilde	74, 1191	\thesection	440, 1194
\textbf	3, 34–39, 644, 677, 690, 1191	\thicklines	718, 1194
\textcolor	107, 1191	\thispagestyle	226, 272, 285, 297, 1194
textcomp package	314, 316, 320	\thisproblem	639, 650, 657, 670, 706,
\textdollar	314, 1192	1195	
textenum environment	645, 1192	\thisproblemlabel	657, 1195
\texteuro	316, 317, 1192	\thr@@	447
\TextField	873, 877, 1192	\tick	967, 969, 1195
\textheight	764, 768, 1192	\ticket	725, 729, 1195
\textifsymbol	832, 1192	ticket definition files (.tdf)	722
\textsubscript	986, 1192	ticket package	279, 708, 722
\textsuperscript	986, 1193	\ticketdefault	724, 726, 729, 1195

Index

		Symbols
\ticketDistance	724, 729, 1195	A N
\ticketNumbers	722, 728, 1195	B O
\ticketSize	724, 728, 1196	C P
\ticknum	967, 1196	D Q
tikz library		E R
arrows.meta	898, 907	F S
calc	747, 1059	G T
plotmarks	964, 971	H U
positioning	898	I V
shadings	832	J W
shadows	832	K X
shapes.geometric	898	L Y
shapes.multipart	540	M Z
tikz package	62, 423, 523, 526, 545, 556, 569, 746, 748, 832, 896, 898, 913, 926, 931, 948, 960, 964	U
tikzpicture environment	523, 537, 909, 910, 926, 928, 945, 962, 1196	UK FAQ 1, 1056
\time	499, 1196	ulem (changes) 985
\timefmt	505, 510	ulem package 985
\timezonefmt	507, 511	\unframedsolutions 625, 1198
\title	572, 1196	\unitlength 710, 712, 713, 716, 717, 719, 720, 722, 724, 728, 734, 735, 1198
title (KOMA variable)	237	units
\titledquestion	610, 616, 1196	sp (scaled point) 55
\titlegraphic	571, 573, 592, 1196	\url 380, 1198
to (isodoc)	262	

Index

			Symbols
url package	380	\verbatiminput	673, 1200
\use@elment	850	vertical (databar)	940
\usecolortheme	590, 1199	\vfill	289, 295, 736, 738, 769, 1200
usedefaultargs (probsln)	639, 657	Vote environment	406, 1201
useenvlab (newlfm)	298	\vote	406, 407, 1201
\usefonttheme	591, 1199	\vspace	610–612, 620, 1201
\useinnertheme	591, 1199		
\useoutertheme	591, 1199		
\usepackage	7, 83, 168, 313, 493, 498, 1199	W	
\useproblem	650, 655, 657, 1199	wasy sym package	735, 810, 822, 832
useregional (datetime2)	464	watermark package	423
\usettheme	590, 1199	website (isodoc)	265
\usetikzlibrary	540, 746, 832, 898, 909, 964, 1199	whitespace	22, 753, 1056, 1057
UTF-8	72, 117, 127, 1056	who (isodoc)	261
utf8 (inputenc)	72	Wine	878
		\write18	108
		X	
\value	56, 367, 815, 817, 818, 1200	x11names (xcolor)	748
vatno (isodoc)	307	\xappto	50, 1201
vc package	992	\xblackout	414, 416, 1201
\vector	711, 712, 1200	\XBox	822, 829, 1201
\verb	575, 1200	xcolor (changes)	985
verbatim environment	575, 1200	xcolor package	424, 428, 545, 748, 901, 985
verbatim package	673	x11names	748
		\xdef	41–44, 557, 561, 567, 663, 1202

Index

- \xDTLassignfirstmatch 196, 202, 272,
276, 296, 301, 304, 1202
\xifinlist 175, 1202
\xifinlistcs 176, 1202
\xpreto 52, 1202
xwatermark package 423, 428, 429, 433
 disablegeometry 429
 picontoptext 428
 printwatermark 428
 showpagecenter 428
- Y**
- \year 470, 482, 498, 501, 663, 690, 700,
702, 1202
yourletter (isodoc) 263
yourref (KOMA variable) 232, 233, 237
yourref (isodoc) 263
- Z**
- zip (isodoc) 261

Symbols	
A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

GNU FREE DOCUMENTATION LICENSE

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus

such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright

notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version

filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as

Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

GNU Free Documentation License

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with … Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

HISTORY

30th September 2015 (Version 1.1)

Corrected missing braces in \ifdate in calendar examples.

30th September 2015 (Version 1.0)

Initial version.

BACK COVER TEXT

(See <http://www.gnu.org/licenses/fdl-howto-opt.html#SEC2>.)

If you choose to buy a copy of this book, Dickimaw Books asks for your support through buying the Dickimaw Books edition to help cover costs.