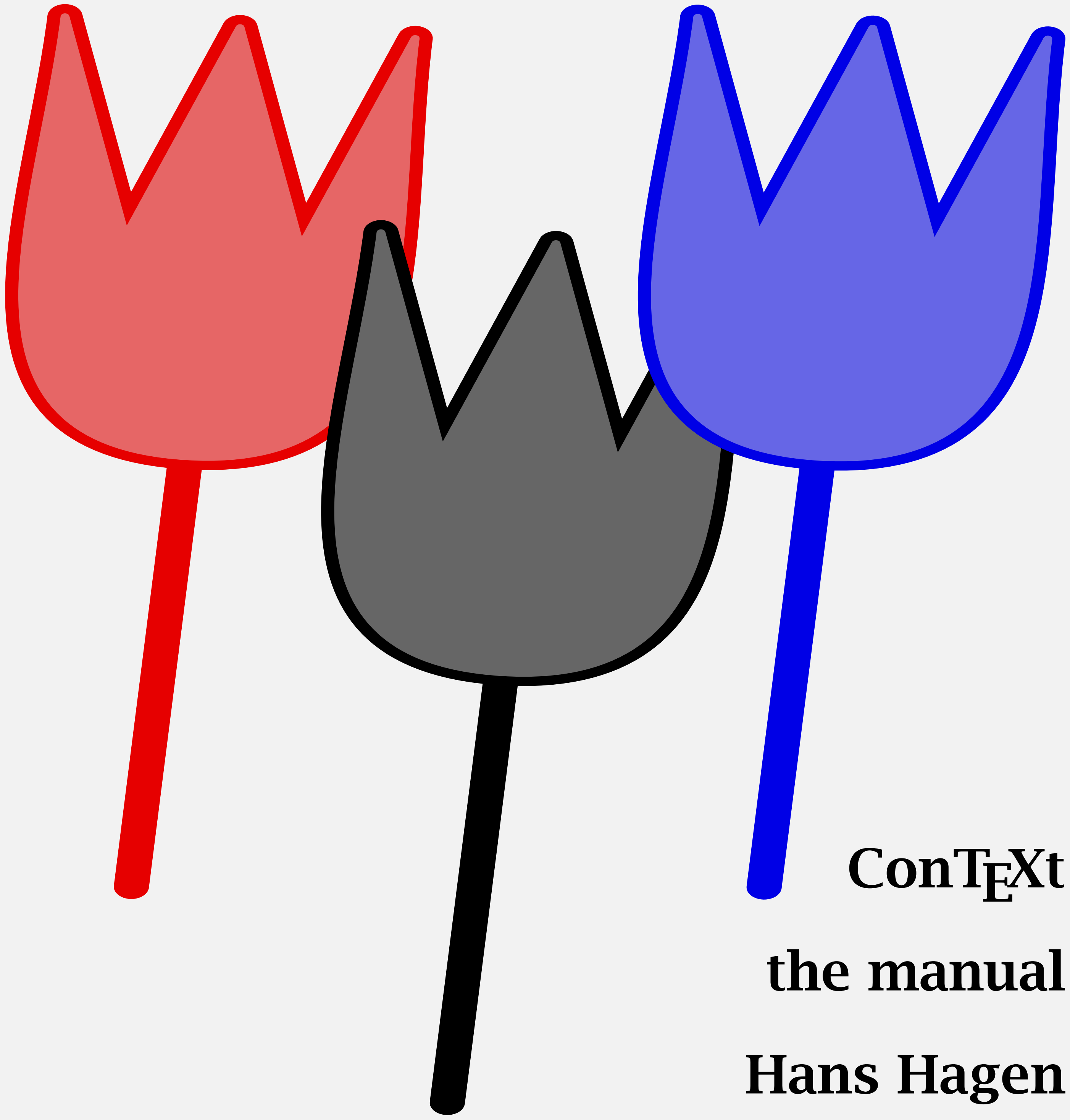


ConTEXt

the manual

Hans Hagen

ConTEXt

the manual

Hans Hagen

January 27, 2000

This document is typeset using PDFTEX. The body font is Lucida Bright at 10 points. The layout is based on a 24 cm square paper size, scaled down .875 to fit nicely on an A4.

Content

Preface

This manual is about ConTEXt, a system for typesetting documents. Central element in this name is the word TEX because the typographical programming language TEX is the base for ConTEXt.

People who are used to TEX will probably identify this manual as a TEX document. They recognise the use of \. One may also notice that the way pararaphs are broken into lines is often better than in the avarage typesetting system.

In this manual we will not discuss TEX in depth because highly recommended books on TEX already exist. We would like to mention:

1. the unsurpassed *The TEXBook* by Donald E. Knuth, the source of all knowledge and TEXnical inspiration,
2. the convenient *TEX by Topic* by Victor Eijkhout, the reference manual for TEX programmers, and
3. the recommended *The Beginners Book of TEX* by Silvio Levy and Raymond Seroul, the book that turns every beginner into an expert

For newcomers we advise (3), for the curious (1), and for the impatient (2). ConTEXt users will not need this literature, unless one wants to program in TEX, uses special characters, or has to typeset math. Again, we would advise (3).

You may ask yourself if TEX is not one of the many typesetting systems to produce documents. That is not so. While many systems in eighties and nineties pretended to deliver perfect typographical output, TEX still does a pretty good job compared to others.

TEX is not easy to work with, but when one gets accustemed to it, we hope you will appreciate its features,

Hans Hagen, 1996–1999

Introduction

1

1.1 TEX

TEX was developed at the Stanford University during the seventies. The designer, developer and spiritual father of TEX is Donald E. Knuth. Knuth developed TEX to typeset his own publications and to give an example of a systematically developed and annotated program.

The TEX project was supported by the American Mathematical Society and resulted in the programming language and program TEX, the programming language and program METAFONT, the Computer Modern typefaces and a number of tools and publications.

TEX is used worldwide, supports many languages, runs on almost every platform and is stable since 1982, which is rather unique in today's information technology.

TEX is a batch–oriented typesetting system. This means that the complete text is processed from beginning to end during which typesetting commands are interpreted. Because you tell your typesetting intentions to TEX, the system can also be qualified as an intentional typesetting system.

In most documents one can stick to commands that define the structure and leave the typographic details to CONTEXT. One can concentrate on the content, instead of on makeup; the author can concentrate on his reader and his intentions with the text. In this respect one can classify CONTEXT as an intentional system. We prefer such a system over a page–oriented system, especially in situations where you have to process bulky documents of with regularly changing content. Furthermore an intentional typesetting system is rather flexible and makes it possible to change layout properties depending on its application. It can also cooperate quite well with other text–processing programs and tools.

1.2 ConTEXt

The development of CONTEXT was started in 1990. A number of TEX based macropackages had been used to our satisfaction. However, the non–technical users at our company were not accustomed to rather complex and non–Dutch interfaces. For this reason we initiated the development of CONTEXT with a parameter driven interface and commands that are easy to understand. Initially the user interface was only available in Dutch.

The functionality of CONTEXT was developed during the production of a great number of complex educational materials and workplace manuals and handbooks. In 1994 the package was stable enough to warrant a Dutch user manual. Over the years ConTEXt has been upgraded with many features and German and English interfaces were added. Though ConTEXt is as (un)stable as any other macropackage there are still a great number of wishes. These will be implemented in the spirit of the existing ConTEXt commands.

ConTEXt comes with a number of Perl scripts, like TEXutil and TEXexec. Also a number of modules are available, like ppchTEX for typesetting chemical structures.

1.3 Commands

A ConTEXt document is normally coded in ascii. Such a document consist of text mixed with ConTEXt commands. These commands tell the system how the text should be typeset. An example of such a command is `\sl`. A ConTEXt command begins with a backslash (\). Most of the time a command does something with the text that comes after the command. The text after the command `\sl` will be typed *slanted*.

When we use a command like `\sl` we are typesetting. Typesetting and writing are conflicting activities. As an author you would rather spend as little time as possible with typesetting. However, you want to indicate that something has to happen with the text. An example is a command like `\em` (*emphasis*). Commands like `\em` enable the typesetter to change the meaning of this command without having to edit the text.

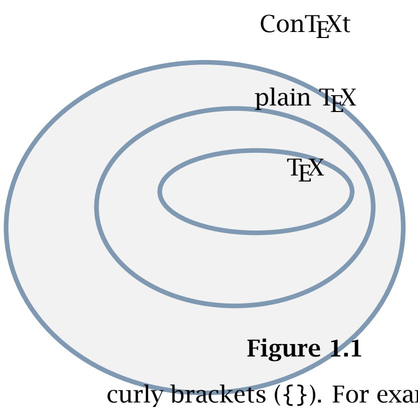


Figure 1.1

A TEX user normally speaks of macros instead of commands. A macro is a, often small, program. Although we will use both ‘command’ and ‘macro’, we try to consistently use the word command for users and macro for programmers.

A collection of macros is called a macropackage. We believe ConTEXt is one of the most extensive and complete macropackages. One of the advantages of ConTEXt is the availability of most of the plain TEX macros.

A command is often followed by setups or by text. Setups are placed between brackets ([]). The scope or range of the command, the text acted upon, is placed between curly brackets ({}). For example:

```
\framed[width=2cm,height=1cm]{that’s it}
```

If we process this text and command by TEX we get:

that's it

Setups in ConTEXt are defined by commands like:

```
\setupsomething[variable=value,variable=value,...]
```

or

```
\setupsomething[option,option,...]
```

In both examples the setups are placed between []. Several setups are defined in a comma list. A number of examples are:

```
\setupwhitespace[big]
\setupitemize[packed,columns]
\setuplayout[backspace=4cm,topspace=2.5cm]
```

There are also commands that are used to define new commands. For example:

```
\definesomething[name]
```

Sometimes a definition inherits its characteristics from another (existing) one. In those situations a definition looks like:

```
\definesomething[clone][original]
```

In many cases one can also pass settings to these commands. In that case a definition looks like:

```
\definesomething[name][variable=value,...]
```

These setups can also be defined in a later stage with:

```
\setupsomething[name][variable=value,...]
```

An example of such a name coupled definiton and setup is:

```
\definehead[section][chapter]
\setuphead[section][textstyle=bold]
```

The alternatives shown above are the most common appearances of the commands. But there are exceptions:

```
\defineenumeration[Question][location=inmargin]
\useexternalfigure[Logo][FIG-0001][width=4cm]
\definehead[Procedure][section]
\setuphead[Procedure][textstyle=slanted]
```

After the first command the newly defined command `\Question` is available which we can use for numbered questions and to place numbers in the margin. With the second command we define a picture that is scaled to a `width` of 4cm. After the third command a new command `\procedure` is available that inherits its characteristics from the predefined command `\section`. The last command alters the characteristics of the newly defined head. Later we will discuss these commands in more detail.

Many typographical operations are performed on a text that is enclosed within a `start-stop` construction:

```
\startsomething
............................
\stopsomething
```

Often keywords or key–value pairs can be passed, that inform ConTeXt on the users wishes:

```
\startnarrower[2*left,right]
............................
\stopnarrower
```

or

```
\startitemize[n,broad,packed]
\item ............................
\item ............................
\stopitemize
```

We use `begin-end` constructions to mark textblocks. Marked textblocks can be typeset, hidden, replaced or called up at other locations in the document.

```
\beginsomething
............................
\endsomething
```

These commands enable the author to type questions and answers in one location and place them at another location in the document. Answers could be placed at the end of a chapter with:

```
\defineblock[Answer]
\setupblock[Answer][bodyfont=small]
\hideblocks[Answer]
............................
\chapter{........}
```

```
............................
\beginofAnswer
............................
\endofAnswer
............................
```

In this case answers will be typeset in a smaller bodyfont size, but only when asked for. They are hidden by default, but stored in such a way, that they can later be typeset.

Commands come in many formats. Take for example:

```
\placefigure
  [left]
  [fig:logo]
  {This is an example of a logo.}
  {\externalfigure[Logo]}
```

This command places a picture at the left hand side of a text while the text flows around the picture. The picture has a reference `fig:logo`, i.e. a logical name. The third argument contains the title and the fourth calls the picture. In this case the picture is a figure defined earlier as `Logo`. Figure 1.1 is typeset this way.

The last example has arguments between optional brackets (`[]`). Many commands have optional arguments. In case these optional arguments are left out the default values become operative.

You may have noticed that a spacy layout of your ASCII text is allowed. In our opnion, this increases readability considerably, but you may of course decide to format your document otherwise. When the ConTEXt commands in this manual are discussed they are displayed in the following way:

```
\setupfootertexts[.1.][.2.][.3.]                                96
.1.     text margin edge
.2.     text section date mark pagenumber
.3.     text section date mark pagenumber
```

The command `\setupfootertexts`, which we will discuss in detail in a later chapter, has three arguments of which the first is optional. The first argument defaults to `[text]`. Optional arguments are displayed as *slanted* text. Default values are underlined and possible alternatives are typeset *slanted*. In this example *text* means that you can provide any footertext. ConTEXt is able to keep track of the status of information on the page, for instance the name of the current chapter. We call this kind of information *mark*, so the command `\setupfootertexts` accept references to marks, like those belonging to sectioning commands: `chapter`, `section`, etc. The argument `date` results in the current systemdate.

When the setup of some commands are displayed you will notice a ◀▶ in the right hand top corner of the frame. This indicates that this command has a special meaning in interactive or screen documents. Commands for the interactive mode only show solid arrows, commands with an additional functionality show gray arrows.

1.4 Programs

TEX does a lot of text manipulations during document processing. However, some manipulations are carried out by TEXUTIL. This program helps TEX to produce registers, lists, tables of contents, tables of formulas, pictures etc. This program is a PERL script.

Document processing can best be done with TEXEXEC. This PERL script enables the user to use different processing modes and to produce different output formats. It also keeps track of changes and processes the files as many times as needed to get the references and lists right.

1.5 Files

TEX is used with ASCII source files. ASCII is an international standardized computer alphabet. The ASCII file with the prescribed extension `tex` is processed by TEX. During this process TEX produces a file with graphical commands. This file has the extension `dvi`. A machine–specific driver transforms this file into a format that is accepted by photosetters and printers. Usually, POSTSCRIPT drivers are used to produce POSTSCRIPT files.

CONTEXT relies on plain TEX. Plain TEX, CONTEXT and a third package TABLE are brought together in a so called format file. TABLE is a powerful package for typesetting tables. A format file can be recognized by its suffix `fmt`. TEX can load format files rather fast and efficiently.

A `dvi` file can be viewed on screen with a dedicated program. For electronic distribution POSTSCRIPT files can be transformed (distilled) into Portable Document Format (PDF) files. PDF files are of high graphical quality and are also interactive (hyperlinked). CONTEXT fully supports PDFTEX, which means that you can generate PDF output directly.

1.6 Texts

1.6.1 Characters

A TEX text contains ASCII characters. Higher ASCII values to produce characters like ë, ô and ñ can also be used in this version of TEX. Some characters in TEX have a special meaning. These characters can be typeset by putting a \ in front of it. A % is obtained by typing `\%`. If one would type only a `%` the result would be undesirable because TEX interprets text after a `%` as comment that should not be processed. A $ is produced by `\$`. A `$` without a \ indicates the beginning of the mathemathical mode.

1.6.2 Paragraphs

TEX performs its operations mostly upon the text element *paragraph*. A paragraph is ended by `\par` or preferably by an empty line. Empty lines in an ASCII text are preferred because of readability.

1.6.3 Boxes

In this manual we will sometimes talk about boxes. Boxes are the building blocks of TEX. TEX builds a page in horizontal and vertical boxes. Every character is a box, a word is also a box built out of a number of boxes, a line is ...

When TEX is processing a document many messages may occur on the screen. Some of these messages relate to overfull or underful boxes. Horizontal and vertical boxes can be typeset by the TEX commands `\hbox` and `\vbox`. Displacements can be achieved by using `\hskip` and `\vskip`. It does not hurt to know a bit about the basics of TEX, because that way one can far more easilly write his or her own alternatives to, for instance, chapter headers.

1.6.4 Fonts

TEX is one of the few typesetting systems that does math typesetting right. To do so TEX needs a complete fontfamily. This means not only the characters and numbers but also the mathematical symbols. Complete fontfamilies are Computer Modern Roman and Lucida Bright. Both come in serif and sans serif characters and a monospaced character is also available. Other fontfamilies are available.

1.6.5 Dimensions

Characters have dimensions. Spacing between words and lines have dimensions. These dimensions are related to one of the units of table 1.1. For example the linespacing in this document is 14.83998pt.

dimension	meaning	equivalent
`pt`	point	72.27pt = 1in
`pc`	pica	1pc = 12pt
`in`	inch	1in = 2.54cm
`bp`	big point	72bp = 1in
`cm`	centimeter	2.54cm = 1in
`mm`	millimeter	10mm = 1cm
`dd`	didot point	1157dd = 1238pt
`cc`	cicero	1cc = 12dd
`sp`	scaled point	65536sp = 1pt

Table 1.1 Dimensions in TEX.

We will often specify layout dimensions in points or centimeters or milimeters. A point is about .35mm. Most dimensions are rather American. The European Didot point is equivalent to 1/2660m = 3.759398496mm.

Next to the mentioned dimension TEX also uses `em` and `ex`. Both are font dependant. An `ex` has the height of an x, and an `em` the width of an M. In the Computer Modern Roman typefaces, numbers have a width of 1/2em, while a — (`---`) is one em.

1.6.6 Error messages

While processing a document, TEX generates status messages (what is TEX doing), warning messages (what could TEX do better) and error messages (what considers TEX wrong). An error message is always followed by a `halt` and processing will be stopped. A linenumber and a `?` will appear on screen. At the commandline you can type `H` for help and the available commands will be displayed.

Some fatal errors will lead to an * on the screen. TEX is expecting a filename and you have to quit processing. You can type `stop` or `exit` and if that doesn't work you can always try `ctrl-z` or `ctrl-c`.

1.7 Version numbers

TEX was frozen in 1982. This meant that no functionality would be added from that time on. However, exceptions were made for the processing of multi–language documents, the use of

8-bits ASCII–values and composed characters. Additionally some bugs were corrected. At this moment TEX version 3.141592 is being used. The final TEX version number will be π, while METAFONT will become the Euler number e.

CONTEXT can handle both ε-TEX and PDFTEX, which are extensions to TEX. Both are still under development, so we suggest using the latest versions available. This manual is typeset using PDF-ε-TEX, with ε-TEX version 2.1 and PDFTEX version 14e.

CONTEXT is still under development. Macros are continually improved in terms of functionality and processing speed. Improvements are made within existing macros. For example the possibility to produce highly interactive PDF documents has altered some low–level functionality of CONTEXT but did not alter the interface. We hope that in due time CONTEXT will be a reasonable complete document processing system, and we hope this manual shows enough of its possibilities. This document was processed with version 2000.1.26.

1.8 Top ten

A novice user might be shooed away by the number of CONTEXT commands. Satisfying results can be obtained by only using the next ten groups of commands:

1. `\starttext`, `\stoptext`
2. `\chapter`, `\section`, `\title`, `\subject`, `\setuphead`, `\completecontent`
3. `\em`, `\bf`, `\cap`
4. `\startitemize`, `\stopitemize`, `\item`, `\head`
5. `\abbreviation`, `\infull`, `\completelistofabbreviations`
6. `\placefigure`, `\externalfigure`, `\useexternalfigures`
7. `\placetable`, `\starttable`, `\stoptable`
8. `\definedescription`, `\defineenumeration`
9. `\index`, `\completeindex`
10. `\setuplayout`, `\setupfootertexts`, `\setupheadertexts`

1.9 Warning

CONTEXT users can define their own commands. These newly defined commands may conflict with plain TEX or CONTEXT commands. Therefore it is advisable to use capital characters in your own command definitions.

```
\def\MyChapter#1%
  {\chapter{#1}\index{#1}}
```

This command starts a new chapter and defines an index entry with the same name.

Documents

2

2.1 Introduction

Why should one use TEX in the first place? Many people start using TEX because they want to typeset math. Others are charmed by the possibility of separating content and make-up. Yet another kind of user longs for a programmable system. And let us not forget those users that go for quality.

When using TEX one does not easily run into capacity problems. Typesetting large documents with hundreds of pages is typically a job for TEX. If possible, when coding a document one should look beyond the current document. These days we see documents that were originally typeset for paper being published in electronic format. And how about making a stripped version of a 700 page document? A strict separation between content and layout (make-up) on the one hand and an acceptable redundancy in structure on the other is often enough to guarantee multiple use of one document source.

A system like CONTEXT is meant to make life easier. When coding a document the feeling can surface that "this or that should be easier". This feeling often reflects the truth and the answer to the question can often be found in this manual, although sometimes obscured. It takes some time to learn to think in structure and content, certainly when one is accustomed to mouse driven word processors. In this chapter we focus on the structure of collections of documents.

2.2 Start and stop

In a self contained text we use the following commands to mark the begin and end of a text:

```
\starttext
\stoptext
```

The first command takes care of a number of initializations and the last command tells TEX that processing can stop. When this command is left out TEX will display a * (a star) on the command line at the end of the job. TEX will expect a command, for example `\end`.

It is advisable to type the document setups before the `\start`-command, the so called setup area of the document. In this way a clever word-processor can identify where the text starts, and therefore can include those setups when it partially processes the document, given of course that it supports partial processing of files.

In the example below a very simple layout is being used.

```
\starttext

\subject{Introduction}

\unknown\ America has always been a land set firmly not in the past, but
in the future. On a recent visit to England, I found dozens of wonderful
bookstores chock full of the past --- ancient history, rooms full of it,
and great literature in such monumental stacks as to be overwhelming. In
the usual American bookstore, history might occupy a few bookcases; great
literature has its honoured place, but this year’s paperbacks dominate. The
past is not disregarded, but neither does it loom so large and run so deep
in our blood.

\blank

{\bf Greg Bear, introduction to Tangents (1989).}

\stoptext
```

The commands `\starttext...\stoptext` may be nested. Within a text a new text containing `\starttext` and `\stoptext` may be loaded.

2.3 Structure

In this section a structured approach of managing your documents is discussed. For very simple and self containing documents you can use the following approach:

```
\environment this
\environment that

\starttext
... some interesting text ...
\stoptext
```

When you have to typeset very bulky documents it is better to divide your document in logical components. ConTEXt allows you to setup a project structure to manage your texts. You have to know that:

- A group of texts that belong together have to be maintained as a whole. We call this a *project*.
- Layout characteristics and macros have to be defined at the highest level. For this, the term *environment* has been reserved.

- Texts that belong together in a project we call *products*.
- A product can be divided into components, these components can be shared with other products. Components can be processed individually.

Programmable word processors can be adapted to this structure.

A *project*, *environment* , *product* or *component* is started and stopped with one of the following commands:

```
\startproject ...  ... \stopproject
...     file
```

```
\startproduct ...  ... \stopproduct
...     file
```

```
\startenvironment ...  ... \stopenvironment
...     file
```

```
\startcomponent ...  ... \stopcomponent
...     file
```

Before a `\start`-`\stop`-pair commands can be added. When a file is not found on the directory ConTEXt looks for the files on higher level directories. This enables the user to use one or more environments for documents that are placed on several subdirectories.

command	**project**	**environment**	**product**	**componnent**
`\project` *name*			⋆	⋆
`\environment` *name*	(⋆)	(⋆)	(⋆)	(⋆)
`\product` *name*	⋆			(⋆)
`\componentonderdeel` *name*			(⋆)	(⋆)

Table 2.1 The structure commands that can be used in the files that make up a project.

To treat products and components as individual documents, the commands in table 2.1 are used. The commands marked with ⋆ are obligatory and the commands marked with (⋆) are optional. The content is typed before the `\stop` command.

```
\startproject documents

\environment layout

\product  teacher
\product  pupil
\product  curriculum

\stopproject
```

An example of a project file.

```
\startproduct teacher

\project   documents

\component teacher1
\component teacher2

\stopproduct
```

The product `teacher.tex` (a teacher manual) can be defined as shown on the opposite site.

```
\startcomponent teacher2

\project documents
\product teacher

... text ...

\stopcomponent
```

Here we see the component.

In most cases working with only `\starttext` and `\stoptext` in combination with `\input` or `\enviroment` is sufficient. A project structure has advantages when you have to manage a great number of texts. Although it is more obvious to process *products* as a whole, it also enables you to process *components* independently, given that the stucture is defined properly.

In principal a project file contains only a list of products and environments. If you would process the project file all products will be placed in one document. This is seldom wanted. This manual for example has a project structure. Every part is a product and every chapter is a component. There are several environments that are loaded in the main project file.

Schematically the coherence between files could be displayed as illustrated in figures 2.1, 2.2 and 2.3.

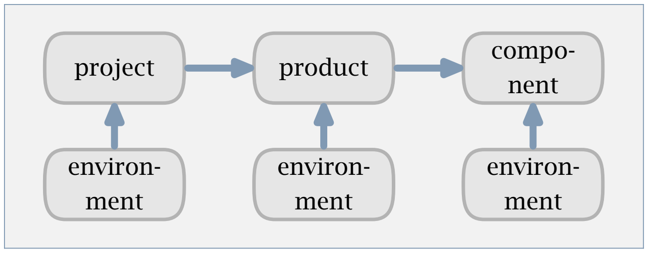


Figure 2.1 An example of project structure.

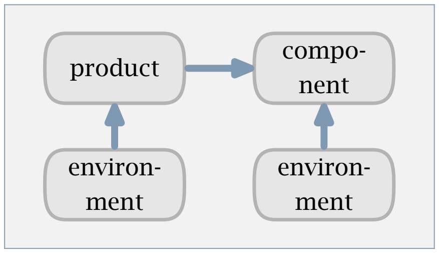


Figure 2.2 An example with only products.

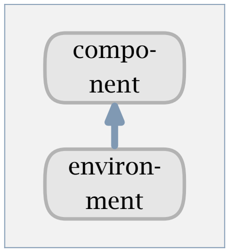


Figure 2.3 An example with only one component.

It is good practice to put all setups in one environment. In case a component or product has a different layout you could define *localenvironments*:

```
\startlocalenvironment[names]
... setups ...
\stoplocalenvironment
```

A local environment can be typed in an environment file or is a separate file itself. When a separate file is used the local environment is loaded with:

`\localenvironment` *name*

Below you will find an example of a project structure.

```
\startproject demos

\environment environ
\product     example

\stopproject
```

file: `demos.tex`

This file is used to define the products and environments.

```
\startenvironment environ

\setupwhitespace[big]

\setupfootertexts[part][chapter]

\stopenvironment
```

file: `environ.tex`

In the environment we type the setups that relate to all the different products. More than one environment or local environments per product can be used.

```
\startproduct example

\project demos

\startintroductions
  \completecontent
\stopintroductions

\startmaintexts
  \component first
  \component second
\stopmaintexts

\startextroductions
  \completeindex
\stopextroductions

\stopproduct
```

file: `example.tex`

The product file contains the structure of the product. Because indexes and registers can be evoked quite easily we do not use a separate file.

```
\startcomponent first

\part{One}

\completecontent

\chapter{First}

..... text .....

\chapter{Second}

..... text .....

\completeindex

\stopcomponent
```

file: `first.tex`

In the components of a product we place the textual content, figures etc. It is also possible to request the tables of content and registers per product.

```
\startcomponent second
\part{Two}
\completecontent
\chapter{Alfa}
..... text .....
\chapter{Beta}
..... text .....
\completeindex
\stopcomponent
```

file: `second.tex`

The product contains more than one component. We could have defined a product for each part and a component for each chapter.

The files `first.tex`, `second.tex` and `example.tex` can be processed separately. As long as there is one product in this project you can also process `project.tex`. If you process an environment there will be no pages of output.

2.4 Directories

Many TEX implementations look for a file in all directories and subdirectories when a requested file is not in the current directory. This is not only time–consuming but may lead to errors when the wrong file (a file with the same name) is loaded.

For this reason CONTEXT works somewhat differently. A file that is not available on the working directory is searched for on the parent directories. This means that environments can be placed in directories that are parents to the products that use them. For example:

```
/texfiles/course/layout.tex
/texfiles/course/teacher/manual.tex
/texfiles/course/student/learnmat.tex
/texfiles/course/otherdoc/sheets.tex
```

The last three files (in different subdirectories) all use the same environment `layout.tex`. So, instead of putting all files into one directory, one can organize them in subdirectories. When a project is properly set up, that is, as long as the project file and specific environments can be found, one can process components and products independently.

2.5 Versions

During the process of document production it is useful to generate a provisional version. This version shows the references and the typesetting failures. The provisional version is produced when you type:

```
\version[...]
...      final concept temporary
```

By default the definitive version is produced. In case a preliminary version is produced the word *concept* is placed at the bottom of each page. The keyword `temporary` shows some information on for instance overfull lines, references, figure placement, and index entries. Most messages are placed in the margin. In some cases these messages refer to the next pages because TEX is processing in advance.

2.6 Modes

TEX can directly produce DVI or PDF. A document can be designed for paper and screen, where the last category often has additional functionality. From one document we can generate different alternatives, both in size and in design. So, from one source several alternatives can be generated.

Processing a file in practice comes down to launching TEX with the name of the file to be processed. Imagine that by default we generate DVI output. Switching to PDF is possible by enabling another output format in the file itself or a configuration file, but both are far from comfortable.

```
\setupoutput[pdftex]
```

for direct PDF output, or for PDF produced from POSTSCRIPT:

```
\setupoutput[dvips,acrobat]
```

The key to the solution of this problem is TEXEXEC. This PERL script provides CONTEXT with a command–line–interface. When we want PDF instead of DVI, we can launch TEXEXEC with:

```
texexec --pdf filename
```

There are more options, like making A5–booklets; more on these features can be found in the manual that comes with TEXEXEC. However, one option deserves more time: modes.

```
texexec --pdf --mode=screen filename
```

The idea behind modes is that within a style definition, at each moment one can ask for in what mode the document is processed. An example of a mode dependant definition is:

```
\startmode[screen]
  \setupinteraction[state=start]
  \setupcolors[state=start]
\stopmode
```

if needed, accompanied by:

```
\startnotmode[screen]
  \setupcolors[state=start,conversion=always]
\stopnotmode
```

One can also pass more than one mode, separated by comma's. There are also some low level mode dependant commands. Given that we are dealing with a screen mode, we can say:

```
\doifmodeelse {screen} {do this} {and not that}
\doifmode     {screen} {do something}
\doifnotmode  {screen} {do something else}
```

A mode can be activated by saying:

```
\enablemode[screen]
\disablemode[screen]
```

Again, we can pass more modes:

```
\enablemode[paper,A4]
```

One strength of TEXEXEC is that one is not forced to enable modes in a file: one can simply pass a command line switch. Just as with choosing the output format: the less we spoil the document source with output and mode settings, the more flexible we are.

2 Modes

Page design

3

3.1 Introduction

While processing a text TEX makes use of the actual `\hsize` (width) and `\vsize` (height). As soon as `\vsize` is exceeded TEX's output routine is launched. The output routine deals with the typeset part — most of the time this will be a page. It takes care of typesetting the headers and footers, the page number, the backgrounds and footnotes, tables and figures. This rather complex process makes it obvious that the output routine actually makes use of more dimensions than `\hsize` and `\vsize`.

3.2 Paper dimensions

With the command `\setuppapersize` the dimensions of the paper being used are defined. There is a difference between the dimensions for typesetting and printing.

```
\setuppapersize[..,.1.,..][..,.2.,..]

.1.     A3 A4 A5 A6 letter ... CD name landscape mirrored rotated 90 180 270
.2.     A3 A4 A5 A6 letter ... name landscape mirrored rotated negative 90 180 270
```

The dimensions of DIN formats are given in table 3.1.

format	size in mm	format	size in mm
A0	841 × 1189	A5	148 × 210
A1	594 × 841	A6	105 × 148
A2	420 × 594	A7	74 × 105
A3	297 × 420	A8	52 × 74
A4	210 × 297	A9	37 × 52

Table 3.1 Default paper dimensions.

Other formats like B0–B9 and C0–C9 are also available. You could also use: `letter`, `legal`, `folio` and `executive`, `envelop 9–14`, `monarch`, `check`, DL and CD.

A new format can be defined by:

```
\definepapersize[...][..,..=..,..]
...       name
width     dimension
height    dimension
offset    dimension
scale     number
```

For example CD was defined as:

```
\definepapersize[CD][width=12cm,height=12cm]
```

After defining CD you can type:

```
\setuppapersize[CD][A4]
```

This means that for typesetting ConTeXt will use the newly defined size CD. The resulting, rather small page, is positioned on an A4 paper size. This second argument is explained in detail later.

ConTeXt can also be used to produce screen documents. For that purpose a number of screen formats are available that relate to the screen dimensions. You can use: S3–S6. These generate screens with widths varying from 300 to 600 pt and a height of 3/4 of the width.

When one chooses another paper format than A4, the default settings are scaled to fit the new size.

3.3 Page composition

In page composition we distinguish the main text area, headers and footers, and the margins (top, bottom, right and left). The main text flows inside the main text area. When defining a layout, one should realize that the header, text and footer areas are treated as a whole. Their position on the page is determined by the topspace and backspace dimensions (see picture 3.1).

The header is located on top of the main text area, and the footer comes after it. Normally, in the header and footer page numbers and running titles are placed. The left and/or right margin are often used for structural components like marginal notes and/or chapter and section numbers. **left** The margins are located in the backspace. Their width has *no* influence on the location of the **right** typesetting area on the page.

On the contrary, the height of the header and footer influence the height of the text area. When we talk about the height, we mean the sum of the header, text and footer areas. When one occasionally hides the header or footer, this guarantees a consistent layout.

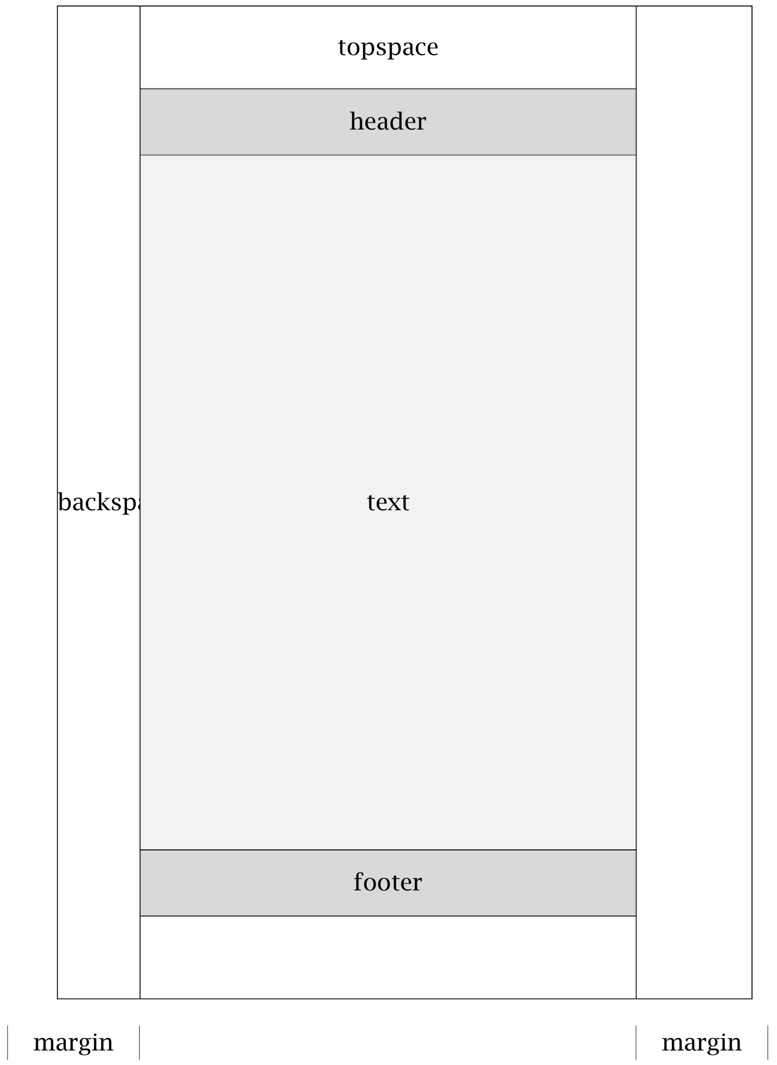


Figure 3.1 The A4 typesetting area and margins (height = header + text + footer).

`\setuplayout[..,..=..,..]`

width	*dimension* fit middle
height	*dimension* fit middle
backspace	*dimension*
topspace	*dimension*
margin	*dimension*
leftmargin	*dimension*
rightmargin	*dimension*
header	*dimension*
footer	*dimension*
top	*dimension*
bottom	*dimension*
leftedge	*dimension*
rightedge	*dimension*
headerdistance	*dimension*
footerdistance	*dimension*
topdistance	*dimension*
bottomdistance	*dimension*
leftmargindistance	*dimension*
rightmargindistance	*dimension*
leftedgedistance	*dimension*
rightedgedistance	*dimension*
horoffset	*dimension*
veroffset	*dimension*
style	normal bold slanted boldslanted type cap small... *command*
marking	on off color
location	left middle right bottom top singlesided doublesided
scale	*dimension*
nx	*number*
ny	*number*
dx	*dimension*
dy	*dimension*
lines	*number*
grid	yes no

The dimensions and location of all those areas are set up with `\setuplayout`.

Setting up the left or right margin has no influence on the typesetting area. In paper documents this parameter is only of use when keywords or other text are placed in the margin (hyphenation).

In paper documents it is sufficient to set up the height, header, footer, top space and back space. In electronic documents and screen documents however we need some room for navigational tools (see chapter ??). In screen documents it is common practice to use backgrounds. Therefore it is also possible to set up the space between the text area and the header and footer on a page, and thereby visually separating those areas.

It is possible to trace the setting by using the following commands:

```
\showframe[...]
...      text margin edge
```

The dimensions can be displayed by:

```
\showsetups
```

A multi-page combination of both is generated with:

```
\showlayout
```

The width of a text is available as `\hsize` and the height as `\vsize`. To be on the safe side one can better use the `\dimen`-registers `\textwidth` and `\textheight`, `\makeupwidth` and `\makeupheight`.

When we are typesetting in one column of text `\textwidth` and `\makeupwidth` are identical. In case of a two columned text the `\textwidth` is somewhat less than half the `makeupwidth`. The `\textheight` is the `\makeupheight` minus the height of the header and footer.

variable	**meaning**
`\makeupwidth`	width of a text
`\makeupheight`	height of a text
`\textwidth`	width of a column
`\textheight`	height – header – footer

Table 3.2 Some `\dimen` variables.

There are also other dimensions available like `\leftmarginwidth` and `\footerheight`, but be aware of the fact that you can only use these variables, you can not set them up. The width of a figure could for instance be specified as `width=.9\leftmarginwidth`.

In principal documents are typeset automatically. However, in some cases the output would become much better if a line would be moved to another page. For these situations you can adjust the layout momentarily (just for that page) by typing:

```
\adaptlayout[..,...,..][..,..=..,..]

...       number
height    dimension max
lines     number
```

The use of these commands should be avoided because if you alter your document the adjustment would not be necessary anymore. So, if you use this command, use it at the top of your document. For example:

```
\adaptlayout[21,38][height=+.5cm]
```

The layout of page 21 and 38 will temporarily be 0.5 cm higher though the footer will be maintained at the same height. The numbers to be specified are the numbers in the output file.

If the layout is disturbed you can reset the layout by:

```
\setuplayout[reset]
```

In some commands you can set up the parameters `width` and `height` with the value `fit`. In that case the width and height are calculated automatically.

On the next pages we will show a number of A5 page layouts centered on an A4. The default setups (dimensions) are adequate for standard documents like manuals and papers. The setup adjusts automatically to the paper size. Notice the use of `middle` while setting up the parameters width and height.

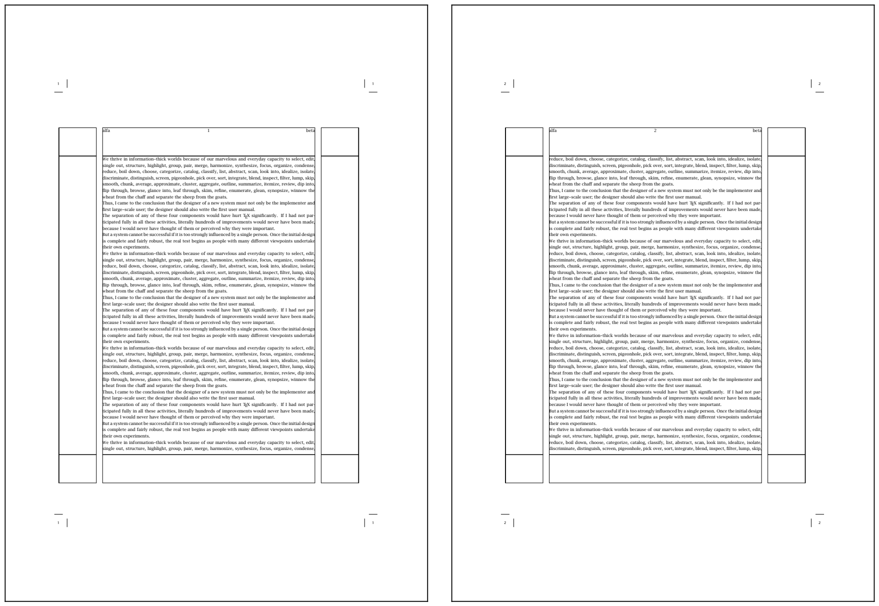


Figure 3.2 The default text-on-page (single sided).

```
\setuppapersize   [A5][A4]
\setuplayout      [location=middle,marking=on]
\setuppagenumber  [alternative=singlesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

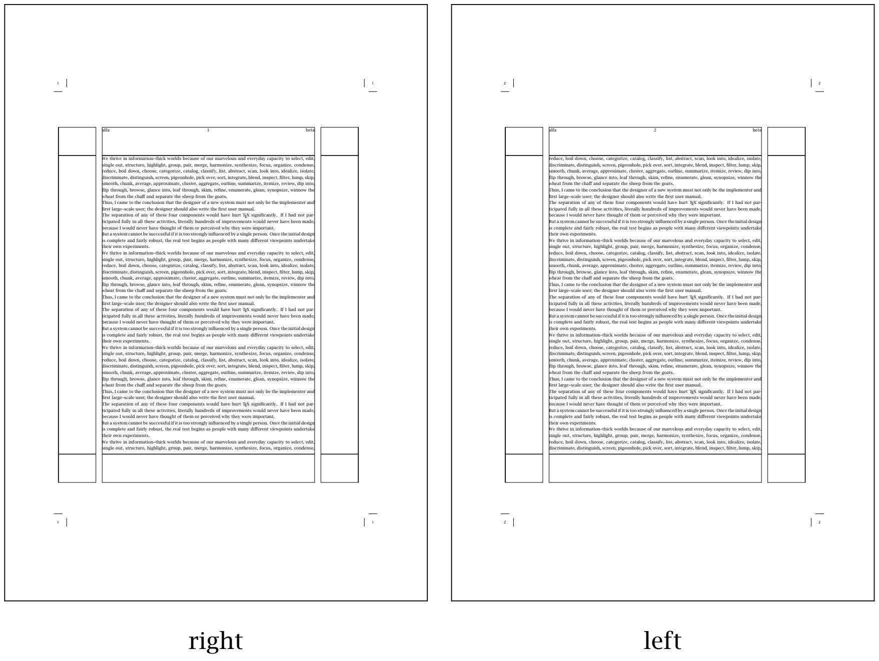


Figure 3.3 The default text–on–page (double sided).

```
\setuppapersize   [A5][A4]
\setuplayout      [location=middle,marking=on]
\setuppagenumber  [alternative=doublesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

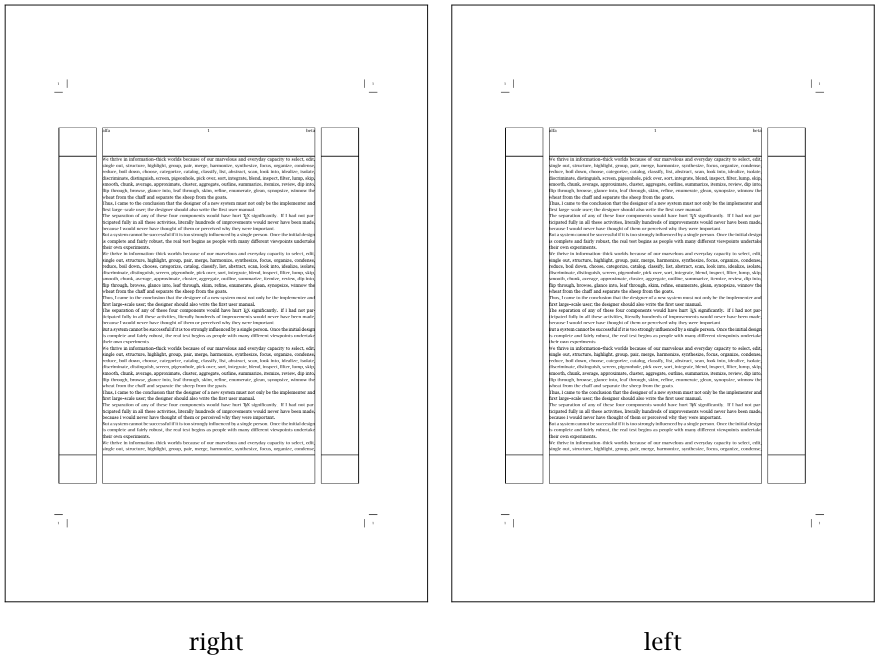


Figure 3.4 The default text–on–page (single–double sided).

```
\setuppapersize   [A5][A4]
\setuplayout      [location=middle,marking=on]
\setuppagenumber  [alternative={singlesided,doublesided}]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta][gamma][delta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

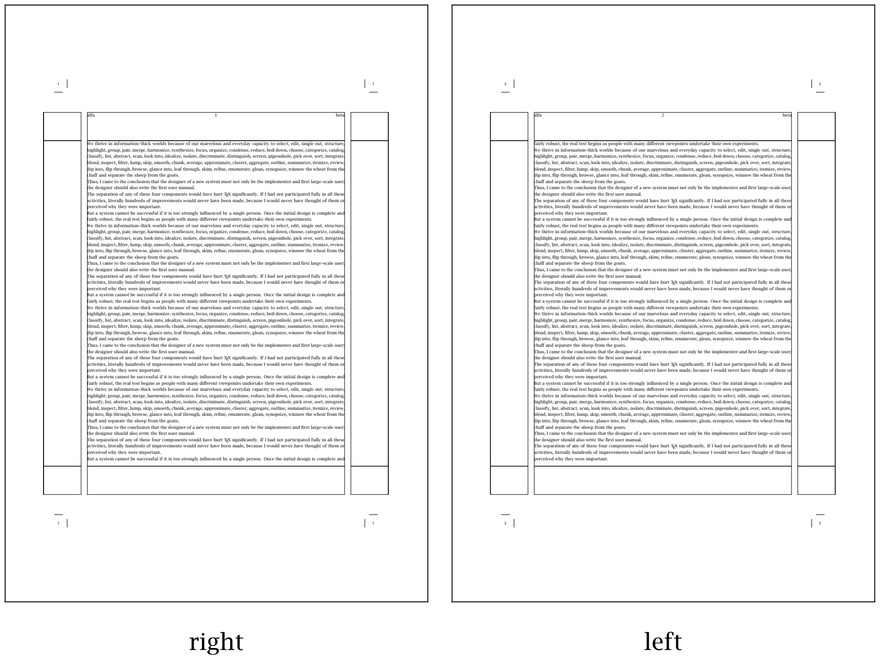


Figure 3.5 Automatically centered text-on-page.

```
\setuppapersize   [A5][A4]
\setuplayout      [backspace=1cm,width=middle,
                   topspace=1cm,height=middle,
                   location=middle,marking=on]
\setuppagenumber  [alternative=doublesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

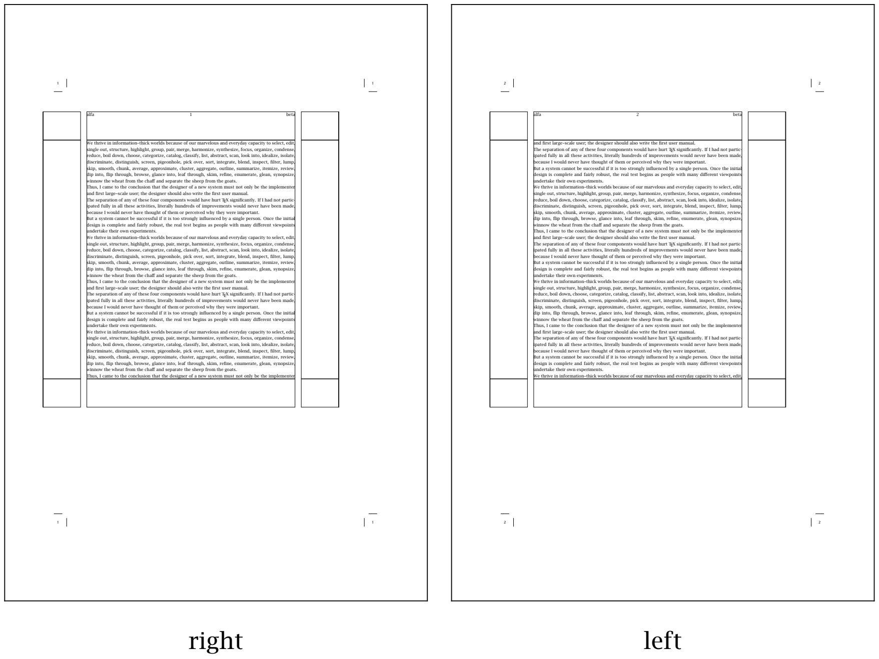


Figure 3.6 A non symmetric text–on–page.

```
\setuppapersize   [A5][A4]
\setuplayout      [backspace=1cm,width=.7\paperwidth,
                   topspace=1cm,height=.7\paperheight,
                   location=middle,marking=on]
\setuppagenumber  [alternative=doublesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

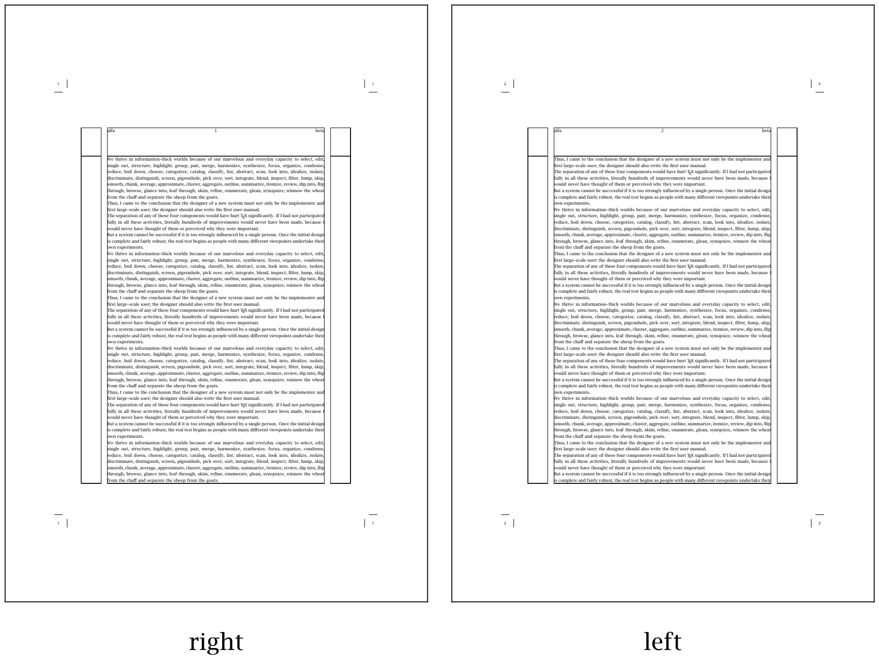


Figure 3.7 A text without `footerheight`.

```
\setuppapersize   [A5][A4]
\setuplayout      [backspace=2cm,width=middle,
                   footer=0cm,margin=1cm,
                   location=middle,marking=on]
\setuppagenumber  [alternative=singlesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\showframe

\starttext
  \dorecurse{10}{\input tufte \par \input knuth \par}
\stoptext
```

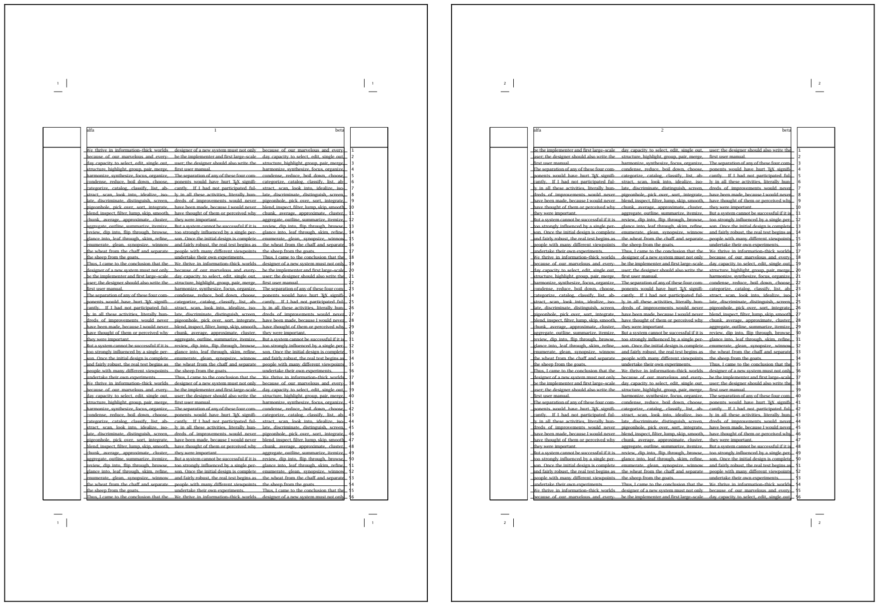
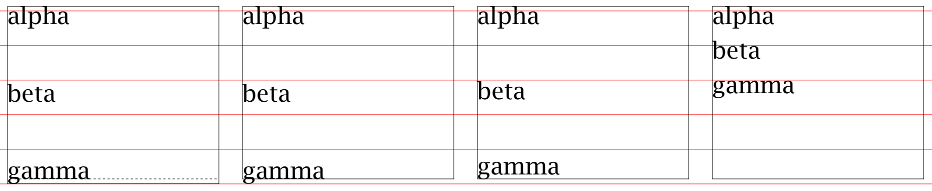


Figure 3.8 A text placed on a grid.

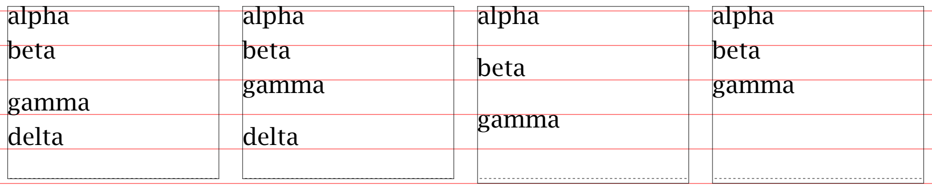
```
\setuppapersize   [A5][A4]
\setuplayout      [location=middle,marking=on]
\setuppagenumber  [alternative=doublesided]
\setupbodyfont    [lbr,6pt]
\setupheadertexts [alfa][beta]

\setuplayout      [headspace=1cm,lines=56,header=1cm,footer=0cm,
                   backspace=1cm,width=middle,grid=yes]

\showframe \showgrid

\starttext
  \startcolumns[n=3]
  \dorecurse{10}{\input tufte \par \input knuth \par}
  \stopcolumns
\stoptext
```

3.4 Grids

There are many ways to align text on a page. Look at the example below and notice the vertical alignment of the words and the white space between the words on the mini pages.



The first three alternatives result in an undesired output. The fourth alternative will lead to pages with unequal length. So we rather make the white space between the lines a little stretchable.[1]



A stretchable line spacing has the disadvantage that lines of two pages or two columns that are displayed close to each other, will seldom align. This is very disturbing for a reader.[2]

In those situations we prefer to typeset on a grid. The means to do this in TEX are very limited but ConTEXt has some features to support grid typesetting.[3]

[2] Here! Another footnote.
[3] Finally, the last footnote!

During typesetting on a grid the heads, figures, formulas and the running text are set on a fixed line spacing. If a typographical component for any reason is not placed on the grid one can snap this component to the grid with:

```
\placeongrid{\framed{This is like a snapshot.}}
```

[1] Hey, watch this. A footnote!

|0.0pt|22.25996pt|7.41998pt|++|4| This is like a snapshot.

This mechanism can be influenced with an argument:

```
\placeongrid[bottom]{\framed{Do you like the snapshot?}}
```

Now an empty line will appear below the framed text. Other parameters are: `top` and `both`. The last parameter divides the linespace between over and below the framed text.

|0.0pt|22.25996pt|7.41998pt|++|5| Now the snapshot looks better.

These examples don't show pretty typesetting. The reason is that `\framed` has no depth because TEX handles spacing before and after a line in a different way than text. CONTEXT has a solution to this:

```
\startlinecorrection
\framed{This is something for hotshots.}
\stoplinecorrection
```

The command `\startlinecorrection` tries to typeset the lines as good as possible and takes the use of grid in account.

|*14.83998pt|14.83998pt|0.0pt|00|6| This is something for hotshots.

Because line correction takes care of the grid we have to use yet another command to stretch the framed text:

```
\moveongrid[both]
\startlinecorrection
\framed{Anyhow it is good to know how this works.}
\stoplinecorrection
```

As you can see this results in somewhat more space:

4.83998pt|22.25996pt|7.41998pt|++|7|

Anyhow it is good to know how this works.

For test purposes one can display the grid with the command `\showgrid`. So grid related commands are:

```
\placeongrid[.1.]{.2.}
.1.      see p 44: \moveongrid
```

|*14.83998pt|44.51994pt|0.0pt|00|8|

```
\moveongrid[...]
...      top both bottom
```

|*14.83998pt|44.51994pt|0.0pt|00|9|

```
\showgrid
```

|*14.83998pt|29.67996pt|0.0pt|00|10|

3.5 Printing

In an earlier section we used page and paper dimensions. In this section we will discuss how these two can be manipulated to yield a good output on paper.

In figure 3.10 and 3.11 we see some alternatives to manipulate the page composition by means of `\setuppapersize` and`\setuplayout`. So it is possible to put a page in a corner or in the middle of the paper, to copy a page and to use cutting marks.

When the parameter papersize is set to `landscape` width and height are interchanged. This is not the same as rotation! Rotation is done by typing `90`, `180` and `270` in the first argument of `\setuppapersize`.

```
\setuppapersize[A5,landscape][A4]
```

These examples don't show that we can correct for duplex printing. For example when we type:

```
\setuppapersize[A5][A4]
\setuplayout[location=middle,marking=on]
```

the front and back side will be placed in the middle of the paper. The markings enable you to cut the paper at the correct size. If we only want to cut twice, we type:

```
\setupppapersize[A5][A4]
\setuplayout[location=duplex]
```

This has the same meaning as `{duplex,left}`. At this setup ConTeXt will automatically move front and back side to the correct corner. In figure 3.9 we show both alternatives.

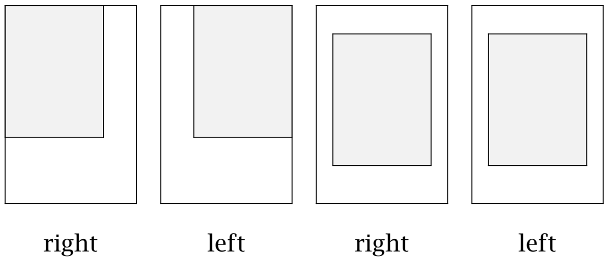


Figure 3.9 Positioning the page on paper for cutting.

Rotating, mirroring, scaling, duplicating and placing pages on paper are independent operations. By combining these operations the desired effects can be reached. Rotating and mirroring and page and paper size are set up at the same time. The other operations are set up with `\setuplayout`.

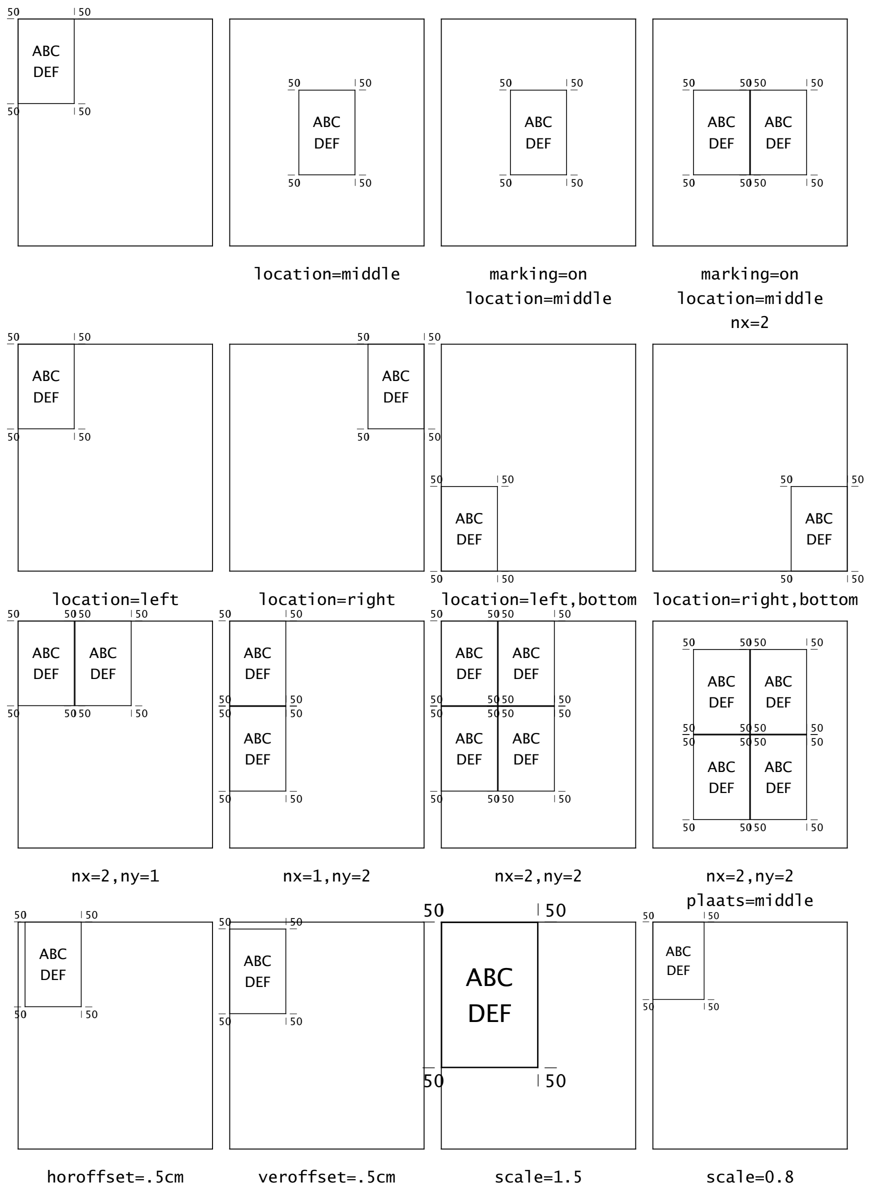


Figure 3.10 Manipulating the page composition with `\setuplayout`.

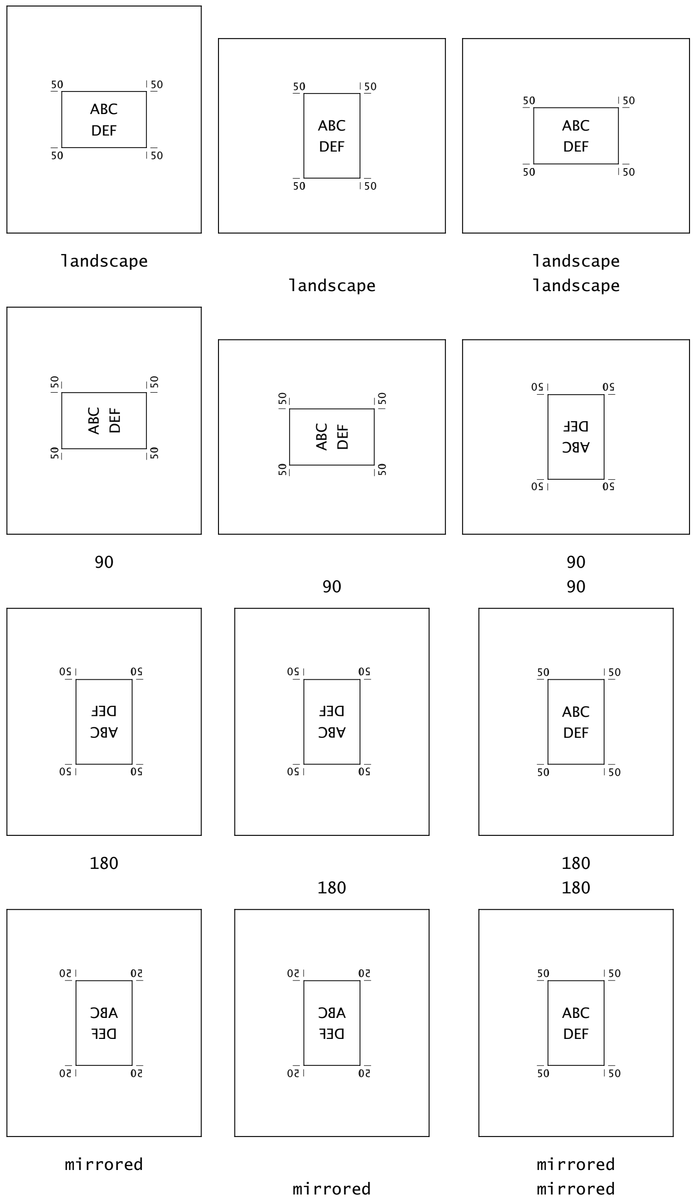


Figure 3.11 Manipulating the page composition with `\setuppapersize`.

```
\showprint[..,.1.,..][..,.2.,..][..,..=..,..]
..=..    see p 29: \setuppapersize
..=..    see p 29: \setuppapersize
..=..    see p 32: \setuplayout
```

You can use `\showprint` to get an idea of how your print will look. However, it is just a representation of the real page as is shown in the examples above.

`\showprint[mirrored][90][location=middle]`

3.6 Arranging pages

By means of `\setuplayout` one can arrange pages on a sheet of paper. A special arrangement for example is that for booklets.

```
\setuparranging[..,...,..]
...      disable 2*16 2*8 2*4 2*2 2UP 2DOWN mirrored rotated doublesided negative 90 180 270
```

We will show some page arrangements on the next pages. If you want to understand how it really works you should try this yourself one day.

The next examples show the cooperation of the commands `\setuppapersize`, `\setuplayout` and `\setuparranging`. Notice how these tests were generated.

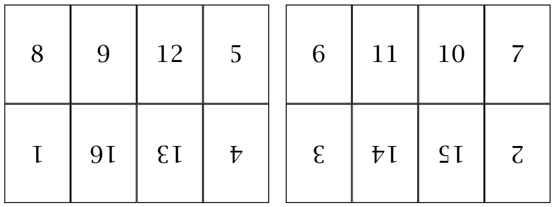


Figure 3.12 The 2*8 arrangement.

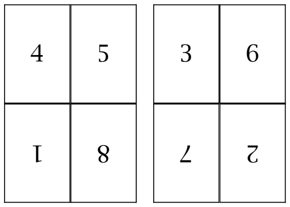


Figure 3.13 The 2*4 arrangement.

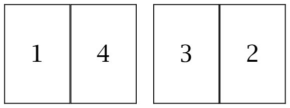


Figure 3.14 The 2*2 arrangement.

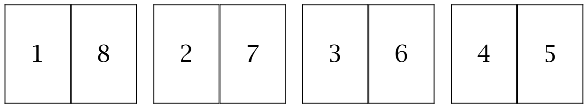


Figure 3.15 The 2UP arrangement.

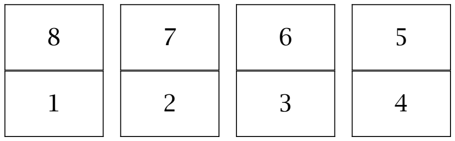


Figure 3.16 The 2DOWN arrangement.

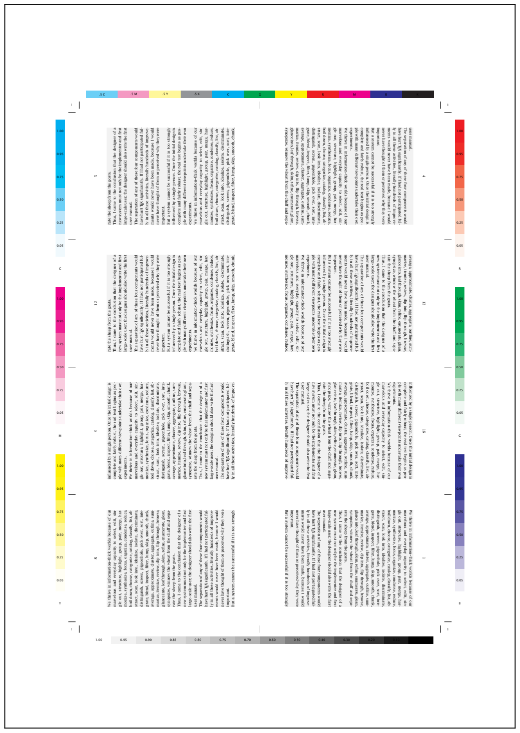


Figure 3.17 Arranging: 16.

```
\setuppapersize    [A7][A3]
\setuparranging    [2*8,rotated,doublesided]
\setuppagenumber   [alternative=doublesided]
\setuplayout       [margin=0pt,width=fit]
\setupbackgrounds  [text][text][background=screen]
\setupcolors       [state=start]
\setuplayout       [location=middle,marking=color]
\setuptolerance    [tolerant]
\setupbodyfont     [lbr,6pt]

\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```

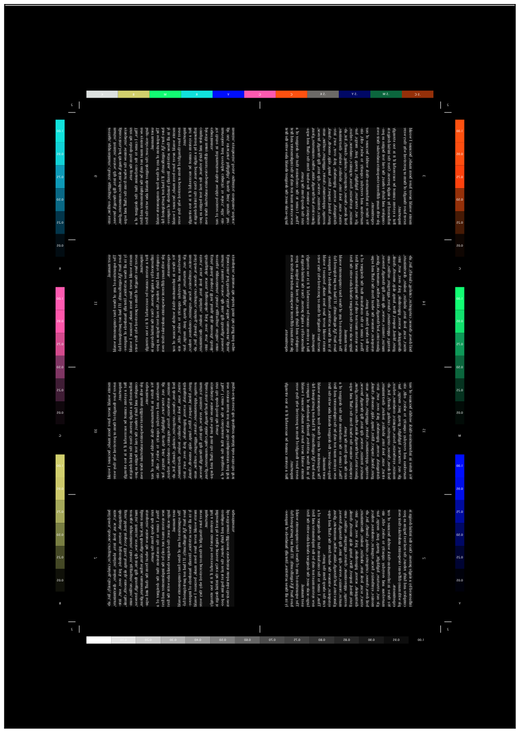


Figure 3.18 Arranging: negative mirrored 16.

```
\setuppapersize    [A7][A3,negative,mirrored]
\setuparranging    [2*8,rotated,doublesided]
\setuppagenumber   [alternative=doublesided]
\setuplayout       [margin=0pt,width=fit]
\setupbackgrounds  [text][text][background=screen]
\setupcolors       [state=start]
\setuplayout       [location=middle,marking=color]
\setuptolerance    [tolerant]
\setupbodyfont     [lbr,6pt]
```

```
\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```

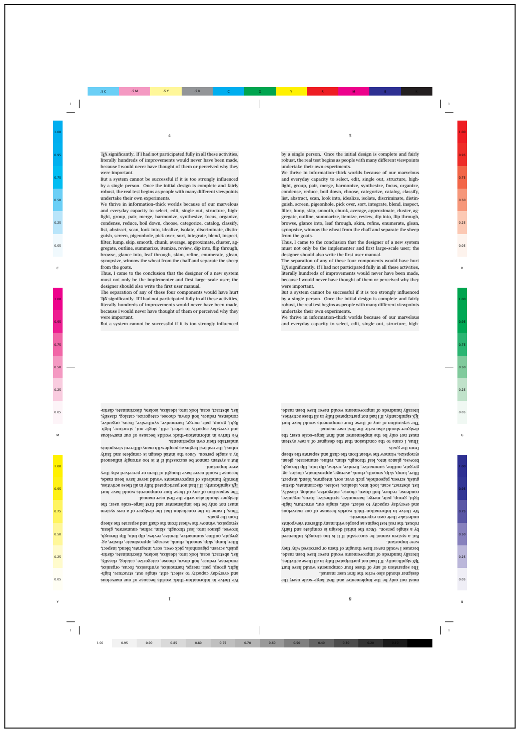


Figure 3.19 Arranging: 8.

```
\setuppapersize    [A6][A3]
\setuparranging    [2*4,doublesided]
\setuppagenumber   [alternative=doublesided]
\setuplayout       [margin=0pt,width=fit]
\setupbackgrounds  [text][text][background=screen]
\setupcolors       [state=start]
\setuplayout       [location=middle,marking=color]
\setuptolerance    [tolerant]
\setupbodyfont     [lbr,7pt]
```

```
\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```

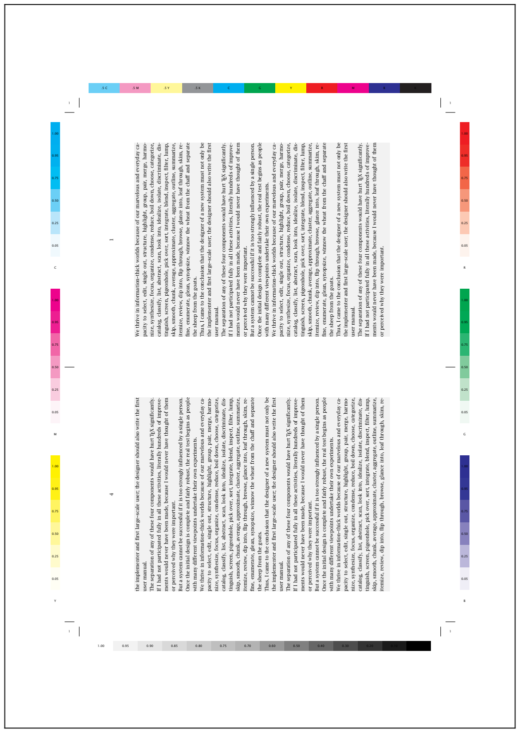


Figure 3.20 Arranging: 4.

```
\setuppapersize    [A5][A3]
\setuparranging    [2*2,rotated,doublesided]
\setuppagenumber   [alternative=doublesided]
\setuplayout       [margin=0pt,width=fit]
\setupbackgrounds  [text][text][background=screen]
\setupcolors       [state=start]
\setuplayout       [location=middle,marking=color]
\setuptolerance    [tolerant]
\setupbodyfont     [lbr,8pt]

\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```

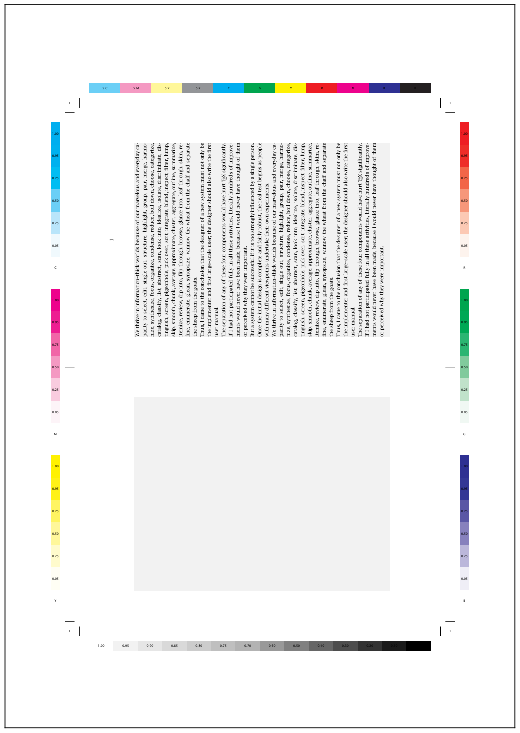
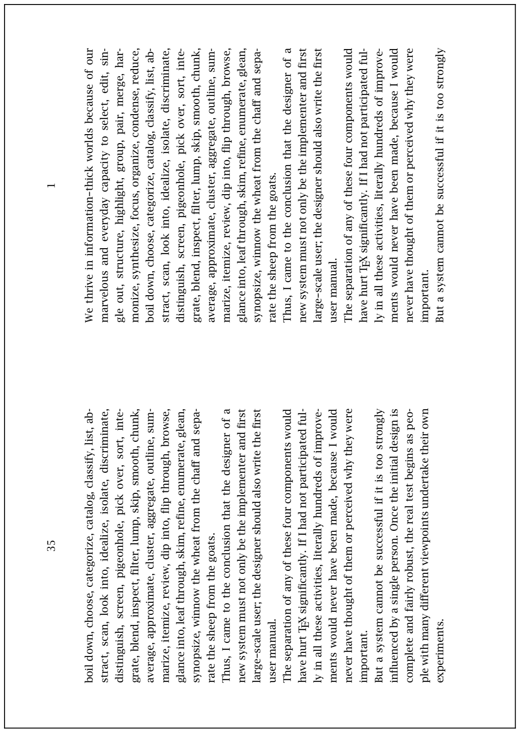


Figure 3.21 Arranging: 2UP (1).

```
\setuppapersize    [A5][A3]
\setuparranging    [2UP,rotated,doublesided]
\setuppagenumber   [alternative=doublesided]
\setuplayout       [margin=0pt,width=fit]
\setupbackgrounds  [text][text][background=screen]
\setupcolors       [state=start]
\setuplayout       [location=middle,marking=color]
\setuptolerance    [tolerant]
\setupbodyfont     [lbr,8pt]
\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```


35

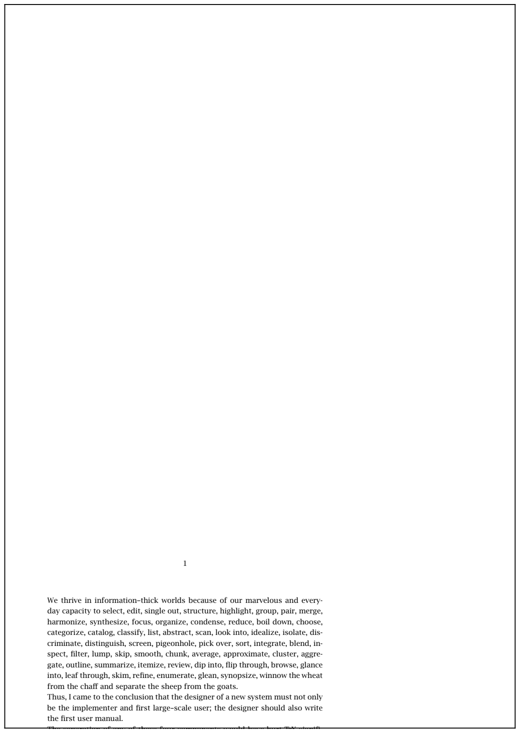
boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

1

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly

Figure 3.22 Arranging: 2UP (2).

```
\setuppapersize   [A5][A4]
\setuparranging   [2UP,rotated,doublesided]
\setuppagenumber  [alternative=doublesided]
\setuplayout      [margin=0pt,width=fit]
\setupbodyfont    [lbr,12pt]

\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```



1

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

Figure 3.23 Arranging: 2DOWN.

```
\setuppapersize   [A4,landscape][A3]
\setuparranging   [2DOWN,doublesided]
\setuppagenumber  [alternative=doublesided]
\setuplayout      [margin=0pt,width=fit]
\setupbodyfont    [lbr,12pt]

\starttext
  \dorecurse{30}{\input tufte \par \input knuth \par}
\stoptext
```

3.7 Logo types

It is possible to place for example company logos at the top or the bottom of a page. We show some examples on the next pages. It is advisable to define a command for typesetting a logo type.

The location of a logo type is defined by:

```
\definelogo[.1.][.2.][.3.][..,..=..,..]

.1.        name
.2.        top header footer bottom
.3.        none page leftedge leftmargin left middle right rightmargin rightedge
command    command text
state      start stop
```

All logo types with `state=start` are automatically typeset on the page. A logo can also be recalled by:

```
\placelogos[..,...,..]

...      name
```

In that case only the listed logos are typeset.

On this page a few potential locations of logos are shown. Temporarily headers and footers of this manual are suppressed. For example the left logo types are defined by means of:

```
\definelogo
  [logo a] [bottom] [left]
  [command=left bottom]
\definelogo
  [logo d] [top] [left]
  [command=left top]
\definelogo
  [logo g] [footer]  [left]
  [command=left footer]
\definelogo
  [logo j] [header] [left]
  [command=left header]
\placelogos[logo a,logo b,logo c,logo d]
```

Instead of `command` we could have chosen `text`. We define the logo with `command` because it is evident that we will use the logo more than once. The example is discussed below.

First we define a command that generates a small logo.

```
\def\ContextLogo%
  {\externalfigure[mp-cont.502][height=24pt,method=mps]}
```

If we want to set this logo at the bottom of every page we type:

```
\definelogo
  [small logo] [bottom] [middle]
  [command=\ContextLogo,state=start]
```

This logo is placed at the bottom of every page. In letters however the logos are located on different positions on the paper. Again, we define the bigger logo including all address information. Watch the use of `\framed`.

```
\def\ContextLetterhead%
  {\hbox
     {\definefont[ContextFont][RegularBold sa 1.5]%
      \ContextFont \setupinterlinespace
      \setupframed
        [align=middle,top=\vfill,bottom=\vfill,
         height=10\bodyfontsize,offset=overlay,frame=off]%
      \framed
        {The\\Con\TeX t\\Chronicle}%
      \externalfigure
        [mp-cont.502][height=10\bodyfontsize]%
      \framed
        {Ridderstraat 27\\8061GH Hasselt NL\\pragma@wxs.nl}}}
```

We also define the position on the paper:

```
\definelogo
  [big logo] [header] [right]
  [command=\ContextLetterhead]
```

This letterhead logo should appear only on the first page. So we simply say:

```
\placelogos[big logo]
```

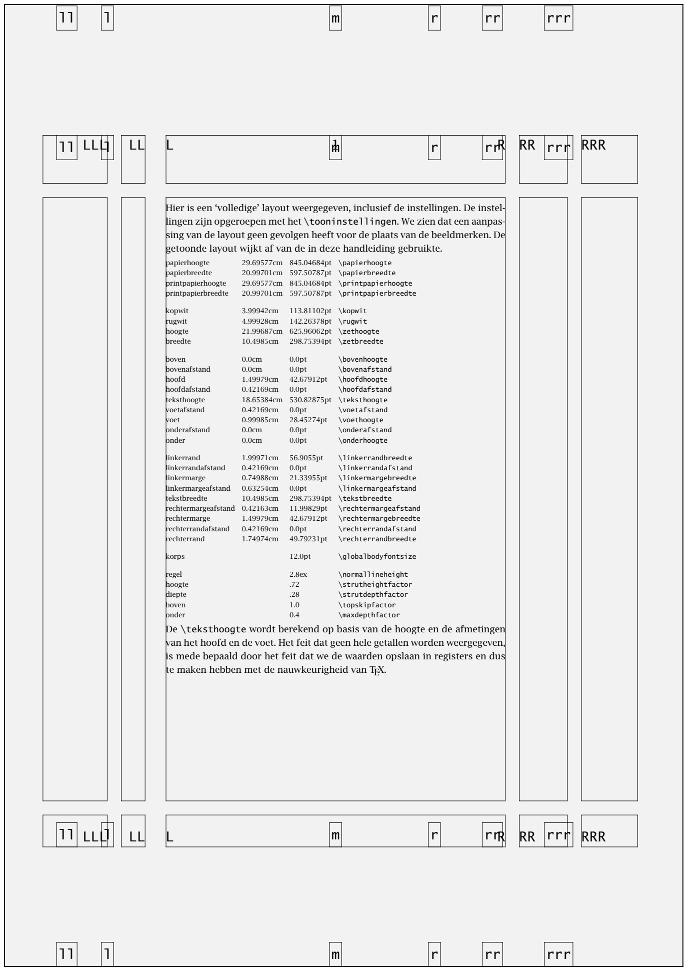
The ConTEXt Chronicle

Ridderstraat 27
8061GH Hasselt NL
pragma@wxs.nl

You will notice that the smaller logo is not placed at the bottom of the page because the command `\placelogos` typesets only the listed logos and suppresses all other logos.

The big logo needs some space on this page so the content of the letter should be moved to a somewhat lower location. We do this with the command:

```
\blank[force,8\bodyfontsize]
```



Hier is een ‘volledige’ layout weergegeven, inclusief de instellingen. De instellingen zijn opgeroepen met het `\tooninstellingen`. We zien dat een aanpassing van de layout geen gevolgen heeft voor de plaats van de beeldmerken. De getoonde layout wijkt af van de in deze handleiding gebruikte.

papierhoogte	29.69577cm	845.04684pt	`\papierhoogte`
papierbreedte	20.99701cm	597.50787pt	`\papierbreedte`
printpapierhoogte	29.69577cm	845.04684pt	`\printpapierhoogte`
printpapierbreedte	20.99701cm	597.50787pt	`\printpapierbreedte`
kopwit	3.99942cm	113.81102pt	`\kopwit`
rugwit	4.99928cm	142.26378pt	`\rugwit`
hoogte	21.99687cm	625.96062pt	`\zethoogte`
breedte	10.4985cm	298.75394pt	`\zetbreedte`
boven	0.0cm	0.0pt	`\bovenhoogte`
bovenafstand	0.0cm	0.0pt	`\bovenafstand`
hoofd	1.49979cm	42.67912pt	`\hoofdhoogte`
hoofdafstand	0.42169cm	0.0pt	`\hoofdafstand`
teksthoogte	18.65384cm	530.82875pt	`\teksthoogte`
voetafstand	0.42169cm	0.0pt	`\voetafstand`
voet	0.99985cm	28.45274pt	`\voethoogte`
onderafstand	0.0cm	0.0pt	`\onderafstand`
onder	0.0cm	0.0pt	`\onderhoogte`
linkerrand	1.99971cm	56.9055pt	`\linkerrandbreedte`
linkerrandafstand	0.42169cm	0.0pt	`\linkerrandafstand`
linkermarge	0.74988cm	21.33955pt	`\linkermargebreedte`
linkermargeafstand	0.63254cm	0.0pt	`\linkermargeafstand`
tekstbreedte	10.4985cm	298.75394pt	`\tekstbreedte`
rechtermargeafstand	0.42163cm	11.99829pt	`\rechtermargeafstand`
rechtermarge	1.49979cm	42.67912pt	`\rechtermargebreedte`
rechterrandafstand	0.42169cm	0.0pt	`\rechterrandafstand`
rechterrand	1.74974cm	49.79231pt	`\rechterrandbreedte`
korps		12.0pt	`\globalbodyfontsize`
regel		2.8ex	`\normallineheight`
hoogte		.72	`\strutheightfactor`
diepte		.28	`\strutdepthfactor`
boven		1.0	`\topskipfactor`
onder		0.4	`\maxdepthfactor`

De `\teksthoogte` wordt berekend op basis van de hoogte en de afmetingen van het hoofd en de voet. Het feit dat geen hele getallen worden weergegeven, is mede bepaald door het feit dat we de waarden opslaan in registers en dus te maken hebben met de nauwkeurigheid van TEX.

Figure 3.24 The location of header, footer, bottom and top logos on a page.

Layout

4

4.1 Introduction

The look of a publication is determined by the page design, the chosen fonts and other aspects like vertical spacing. In this chapter we will explore the latter. Sometimes we will go into detail but a novice user can skip such parts. In normal applications, the default setups are most adequate, because they will adapt to the different situations. For the impatient reader we will just mention a few setups. Spacing between paragraphs is defined by:

```
\setupwhitespace[big]
```

In your source file you can best use an empty line between paragraphs. This increases readability and it makes the typing of `\par` at the end of each paragraph obsolete. Indentation at every new paragraph is obtained by:

```
\setupindenting[medium]
```

A doublesided publication is generated when you type:

```
\setuppagenumbering[alternative=doublesided]
```

As you might expect this might generate page numbering on the right and left hand side of a paper and the margins will be mirrored automatically.

As we have said before only the curious have to read on.

4.2 Paragraphs

The most important unit in TEX is paragraph. A new paragraph is forced by:

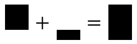
1. an empty line
2. the TEX-command`\par` or `\endgraf`
3. the ConTEXt-command `\paragraph`

The first alternative is the most obvious. You will obtain a readable input file (ASCII file) and errors are minimized. The second alternative is chosen when it is mandatory to the used command. For example in definitions (see 10.2).

4.3 Line spacing

In TEX linespacing is determined by a number of variable dimensions like `\topskip`, `\parskip` and `\baselineskip`. However, in ConTEXt these variables are related to the bodyfont size.

A line has a height and a depth. The distance between two lines is normally equal to the sum of the maximum height and maximum depth:

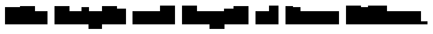


This sum is in ConTeXt equal to 2.8ex, so almost three times the height of an x. This is about 1.2× the bodyfont height. The proportion between maximum height and depth is default .72 : .28. Linespacing alters when a new bodyfont is used or when linespacing is defined explicitly by:

```
\bfd \setupinterlinespace Now, the interline spacing is larger ... \par
```

One has to consider the fact that TeX operates on paragraphs and within a group one has to close the paragraph explicitly with an empty line or `\par`. However, in most cases ConTeXt will take care of this.

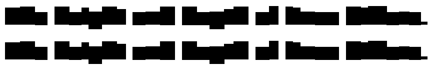
Sometimes a line does not have the maximum height or depth. The next example illustrates this:



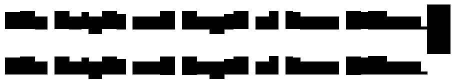
It says:

The height and depth of lines differs.

When we put two of these lines above each other we will get:



You can see that the distance is somewhat bigger that the sum of the height and depth of each separate line. This distance is called the baseline distance (`\baselineskip`) and is in this document 14.83998pt. If we add some extra height to the line we see this:



To prevent the lines from touching TeX adds a `\lineskip`, in our example 1.0pt. In a similar way TeX is taking care of the first line of a page to have at least a height of `\topskip` (here 10.6848pt).

Linespacing is setup by:

```
\setupinterlinespace[...][..,..=..,..]
...       reset small medium big on off
height    number
depth     number
line      dimension
top       number
bottom    number
```

Linespacing adapts to the size of the actual bodyfont automatically. This means that the user can leave this command untouched, unless a different linespacing is wanted. Instead of a factor one of the predetermined values `small` (1.0), `medium` (1.25) or `big` (1.5) can be given. Below an example is given of a text with a linespacing of 1.25: `\setupinterlinespace[medium]`.

Whenever it comes to my mind that “everything that comes in quantities, will somehow survive”, I also got the feeling that in a few hundred years people will draw the saddening conclusion that all those top–ten hits produced by computers represent the summum of todays musical and instrumental abilities. Isn’t it true that archeologists can spend a lifetime on speculating about some old coins from the first century? On the other hand, the mere fact that one can have success with this type of non–music success of some top–hit musicians demonstrates both the listeners inability to rate the product and the lack of self criticism of the performers. In principle the future archeologist will therefore draw the right conclusion.

When you make a fontswitch the linespacing is adapted when you type the command `\setupinterlinespace` without set up parameters or by adding `reset`, for example: `[reset,medium]`.

In books meant for children we often find a somewhat bigger typeface, for instance because we are convinced that this enables them to read the book themselves. On the other hand, I can also imagine that it is a cheap way to increase the number of pages. Unfortunately scaling up will also uncover the lack of quality of the typesetting used and/or the lack of typographic knowledge of the user of such a system. The interline space sometimes differs on a line by line basis, and depends on the height of the current line. Therefore, when changing the style, something that should only be done on purpose, also change the baseline distance when needed.

The text above is typeset in the fontsize `\tfa` which equals `\rma` (see also chapter 5).

```
\start \tfa \setupinterlinespace In books meant for children we often find
a somewhat ... when needed. \par \stop
```

In this example the `\par` is necessary because otherwise TEX will have forgotten the linespacing before the paragraph is finished (in that case, the paragraph is ended by the empty line after the `\stop`).

Instead of a keyword, one can pass a key–value pair to define the characteristics of a line. The default settings are:

```
\setuplinespacing
  [height=.72,
   depth=.28,
   top=1.0,
   bottom=0.4,
   line=2.8ex]
```

The `height` and `depth` determine the ratio between the height and depth of a line. The baseline distance is set to 2.8ex. The parameters `top` and `bottom` specify the relation between the bodyfont size and the height of the first line and the depth of the last line on a page. The last two quantities are related to TEX's `\topskip` and `\maxdepth`.

4.4 Indentation

When a text has little whitespacing, for example in a novel, it is a custom to indent each new paragraph. Indentation is setup with:

```
\setupindenting[..,...,..]
...      none small medium big next first dimension
```

By default there is 'no' indentation. When indentation is turned on, when possible the commands will determine whether indentation is necessary. For example, it doesn't look good to indent after a vertical whitespace. In a number of cases it is even undesirable to indent. Think for example of headers and itemizations.

This manual is typeset without indentation. The great quantity of short sentences and examples would result in a very messy page layout.

When indentation is used, we may have to tell TEX in some cases *not* to indent. This is done by:

```
\noindenting
```

We can set up indenting by:

```
\indenting[..,...,..]
...       never not no yes always first next
```

The meaning of the setups is described in table 4.1. Next to the commands described above we could use the TEX-commands `\indent` and `\noindent`.

setup	**result**
`no` / `not`	don't indent the next paragraph
`yes` / `always`	turn on indentation
`never`	turn off indentation
`first`	indent first paragraphs too
`next`	don't indent first paragraphs

Table 4.1 The way of indenting.

The settings `first` and `next` determine if paragraphs following whitespace should be indented or not. It is a sort of custom not to indent these.

A text my be typeset smaller than the default textwidth. In that case the complete text will be indented on both sides.

```
\startnarrower[..,...,..] ... \stopnarrower
...       n*left n*middle n*right
```

For example:

```
\startnarrower[3*left,2*right]
The relatively small revolution in in Russia in 1917 had big consequences for
this country as well as the rest of the world. It is interesting to see that
some 80~years later a just as small revolution was needed to undo the 1917
```

```
one. In both cases, the main reason for the revolutions was to prevent
democracy to arise.
\stopnarrower
```

Will become:

> The relatively small revolution in in Russia in 1917 had big consequences for this country as well as the rest of the world. It is interesting to see that some 80 years later a just as small revolution was needed to undo the 1917 one. In both cases, the main reason for the revolutions was to prevent democracy to arise.

Next to using `left`, `right` and `middle` also combinations and manifolds are possible. Indentation in the example above could have obtained by typing `2*middle,left`. So, `middle` is equivalent to `left,right`.

The value of indentation is set up by:

```
\setupnarrower[..,..=..,..]

left      dimension
right     dimension
middle    dimension
```

4.5 Vertical spacing (whitespacing)

Vertical spacing between paragraphs is set up by:

```
\setupwhitespace[...]

...      none small medium big line fixed fix dimension
```

Instead of a random value it is better to use one of the pre defined dimension. Default there is no vertical spacing. Without any set up values the vertical spacing is related to the actual fontsize.

Vertical spacing can be forced by either:

```
\whitespace
```

```
\nowhitespace
```

These commands have only effect when vertical spacing is set up. In fact these commands will not be necessary for ConTEXt takes care of most situations.

TEX handles vertical spacing around lines quite different from that around text. In case these problematic situations occur one can use the following commands. Spacing around figures and tables is dealt with by ConTEXt, so only use these commands when the typeset text looks really bad.

```
\startlinecorrection ... \stoplinecorrection
```

For example:

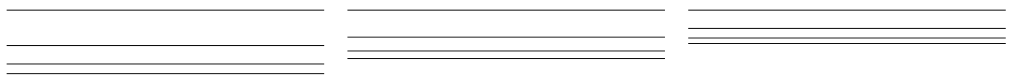
```
\startlinecorrection
\framed{To boxit or not, that's a delicate question!}
\stoplinecorrection
```

One can add vertical spacing with the TEX command `\vskip`, but please don't. We advise you to use:

```
\blank[..,...,..]
...     n*small n*medium n*big nowhite back white disable force reset line halfline formula
        fixed flexible
```

We can use a value of one of the keywords `small`, `medium` or `big`. A big jump is twice a medium jump which is four times a small jump. A value however can be left out (`\blank`) when the default vertical space is desired. It is advisable to set up the vertical spacing only once in the setup area of your document. Local alterations throughout your document will result in a badly-spaced document.

Normally there is some stretch in the vertical spacing. This enables TEX to fill out a page optimally. In the next example we see what happens when we add stretch to whitespace. Each sample shows from top to bottom three `\blank`'s of `big`, `medium` and `small`. The left and right sample show the range of the stretch. The rightmost sample shows that adding stretch can result in shrink.



maximum stretch no stretch minimal stretch

The last vertical space can be undone by typing `\blank[back]` and the next blank can be blocked by `disable`. With `reset` a `disable` is ignored.

The command `\blank` is one of the more advanced commands. The next call is allowed:

```
\blank[2*big,medium,disable]
```

Since `medium` is half the amount of `big`, this results in adding a vertical spaces of 2.5 times `big`. The previous vertical space will be undone automatically and the `disable` suppressed the next `\blank`.

A lasting vertical space can be sustained by `force`. For example, if you want some extra spacing at the top of a page you will have to type `force`.

The default vertical spaces are set up with:

```
\setupblank[...]
...      normal standard line dimension big medium small fixed flexible
```

An example of such a definition is:

```
\setupblank[big]
```

The vertical spaces will be automatically adapted to the fontsize and they are flexible. Changing the default set up locally is therefore not advisable. Without an argument `\setupblank` adapts to the actual fontsize!

The keywords `fixed` and `flexible` are used to end or reinstate this adaptive characteristic. In columns it is recommended to use the setup `[fixed,line]` or the opposite setup `[flexible,standard]`.

This text is typeset a bodyfont of 10pt and is downscaled by a few percent. The setup that is used in this document is shown in table 4.2. We see some stretch in the vertical spacing. The stretching enables TEX to fill out a page satisfactorily. Default the maximal vertical space is 75% of the line space and the stretch maximal of 25%.

setup	value
small	2.78249pt plus 0.92749pt minus 0.92749pt
medium	5.56499pt plus 1.855pt minus 1.855pt
big	11.12997pt plus 3.70999pt minus 3.70999pt
line	14.83998pt

Table 4.2 The whitespace values to a 10pt bodyfont.

In paragraph ?? it was said that the vertical spacing can be set up with the command `\setupwhitespace`. Default there is no whitespace between paragraphs. The setup of vertical spacing and line spacing are related to each other.

Instead of direct setup you can use an indirect way. This has the advantage that you can change the layout more easily. In that case we use:

```
\defineblank[.1.][.2.]

.1.      name
.2.      see p 70: \setupblank
```

If we type for example:

```
\defineblank[aroundverbatim][medium]
```

than `aroundverbatim` is equal to `medium`, which can be used, for example around verbatim, as in:

```
\setuptyping
  [before={\blank[aroundverbatim]},
    after={\blank[aroundverbatim]}]
```

If we want some more whitespacing we only have to change the definition of `aroundverbatim`:

```
\defineblank[aroundverbatim][big]
```

The vertical spacing between two lines can be suppressed with the command:

```
\packed
```

Vertical spacing between more than one line is suppressed by:

```
\startpacked[...] ... \stoppacked
...     blank
```

The spacing around ‘packed’ text is automatically corrected. Opposed to this command is:

```
\startunpacked ... \stopunpacked
```

Skipping more than one vertical space is done with:

```
\godown[...]
...     dimension
```

One of the most important lessons to be learned is to avoid using `\vskip` in running text. This can interfere with some hidden mechanisms of ConTeXt.

Sometimes TeX is not able to sort out spacing on its own. In such situations one can insert the next command at the troublesome location.

```
\correctwhitespace{...}
```

Normally one will not need this command, although sometimes when writing macros, it can be added to make sure that the spacing is okay. Use this kind of tweaking with care!

4.6 Word spacing

Default a space is placed after a period that ends a sentence. In some countries it is custom to stretch the space after a period. Especially documents typeset in small columns will look better that way. Because this is a language specific feature. the default depends on the language. One can however (temporarily) change this spacing.

```
\setupspacing[...]
...     broad packed
```

In many cases we combine words and numbers that should not be separated at linebreaking, for example number 12. These combinations can be connected by a tight space: `number~12`.

Word and number will never be separated at linebreaking on that spot. A space can be made visible by:

```
\space
```

Undesired spaces can be suppressed by:

```
\nospace
```

When you want to align a row of numbers you can use tight spaces with the width of a number. Tight spaces are activated by:

```
\fixedspaces
```

After this command the ~ (tilde) generates a tight space with the width of a number.

4.7 Struts

A strut is a little invisible block without width but with the maximal height and depth of a character or line. If you want to force these maximal dimensions, for example when you are using boxes in your own commands, than you can use the command `\strut`:

`\hbox{\strut test}`

If we leave out the strut in this example the box has no depth. The characters in the word test don't reach under the baseline. Compare for example test (with strut) with test.

Many commands use struts automatically. If for some reason you don't want struts you can try to suppress them by `\setnostrut`. However take care that this command works only locally. A strut can be set by `\setstrut`.

The struts that are used by CONTEXT can be made visible with the command:

```
\showstruts
```

4.8 Text in the margin

Texts can be place in the margins with:

```
\inmargin[.1.][ref]{.2.}
.1.      + - low
.2.      text
```

A new line in a margin text is forced with `\\`. An example of a margin text is:

```
\inmargin{the marginal\\influence of\\advertisement}It would be great
if the recent reduction in washing powder needed to get your wash
perfectly clean had resulted in an equal reduction of time needed to
advertise this kind of products.
```

or:

the marginal influence of advertisement

It would be great if the recent reduction in washing powder needed to get your wash perfectly clean had resulted in an equal reduction of time needed to advertise this kind of products.

over here

When this command is used in the middle of a paragraph the margin text will appear on the same line in themargin. The command `\inmargin` puts the text in the left or right margin. The location where the text will show up depends on the character of the document: single–sided or double–sided. You can also force the text into a specific margin, using:

```
\inleft[.1.][ref]{.2.}
.1.      + - low
.2.      text
```

```
\inright[.1.][ref]{.2.}
.1.      + - low
.2.      text
```

There is also:

```
\inothermargin[.1.][ref]{.2.}
.1.      + - low
.2.      text
```

Some examples of the use of margin text appear below:

```
\startlines
\inleft{to be}\quotation{To be or not to be} to me
\inright{or not}is rather famous english
\inmargin{to be}And just as it is meant to be
that quote will never perish
\stoplines
```

This will become:

to be “To be or not to be” to me
is rather famous english **or not**
And just as it is meant to be **to be**
that quote will never perish

123

The mechanism of margin texts is rather complex. If you think of multiline margin texts and the alignment of these lines with the lines in the textbody you can imagine a few typographic problems. The number 123 next to this paragraph is not aligned but is typeset somewhat lower. This is done by adding the keyword `low`:

```
\inmargin[low]{\ssd 123}The mechanism of margin texts ...
```

It is possible to set up the way margin texts are typeset by means of the command:

```
\setupinmargin[...][..,..=..,..]
...          left right number
location     left right both
style        normal bold slanted boldslanted type cap small... command
before       command
after        command
align        inner outer left right middle normal no yes
line         number
distance     dimension
separator    text
..=..        see p 264: \setupframed
```

a rather marginal effect

With `align` we define the left or right alignment of the margin text. Default margin texts are right aligned. In this example alignment is `middle`.

We can also align on the left of right side automatically. In a double sided document design optimisation of the margin text may ask for more than one processing step. In the example below you see some of the possible setups.

left This is `left` aligned
middle but this goes in the `middle`. Don't forget that
right `right` in this sense, align means a ragged right margin.
yes Just to be complete, there is `yes`
no and `no`.
inner The outsiders `inner` and
outer `outer` adapt themselvs to a doublesided design.

The left and right margin can be set up separately by adding `[left]` or `[right]` as the first argument.

that way we can move quite some text into the margin

With `before` and `after` we can influence margin texts. Bij default the same line spacing is used as in the textbody. But when a narrower fontsize is used we can also adapt the interline spacing. For example:

```
\setupinmargin
  [style=\bfx\setupinterlinespace]
```

Page breaking and margin text are in conflict with each other. The reason is that TEX first typesets a complete page in order to be able to determine the right spot for page breaking. However the margin text is already typeset at that moment. In a next processing stage the margin texts are typeset correctly. If you want to force margin texts in a margin you can type `\inmargin[+]`.

The next command can be compared with the command like `\section`. Before the command is placed in the margin TEX looks if it can be placed on the actual page. If not, it is moved to the following page.

```
\margintext[.1.][ref]{.2.}

.1.     + - low
.2.     text
```

The layout of your ASCII–file will not interfere with the function of this command. This may seem obvious, but TEX programmers know that it is not the case. For example even commands that take care of index entries can be typed close to the margin texts.

The layout of your ASCII–file will not interfere with the function of this command. You might not expect it to, but TEX programmers know that with TEX, the layout of the source usually interferes with for instance margin texts and index entries. In ConTEXt commands that take care of margin texts take care of this situation, so that index entries can be typed close to the margin texts and margin texts can be separated from the next paragraph by an empty line. The same cannot be said for other TEX macropackages.

```
\margintext{text in themargin}
\index{margintexts}
```

```
After experimenting a long time I have succeeded to filter
empty lines and commands that stand between body texts and
margin texts. It is amazing but the index entry really works.
```

Because of the close relation with the page design the margin width is set up by means of: `\setuplayout` (see section 3.3).

The command `\margintext` enables you to put texts in the margin that show completely different characteristics than that of the text body. You can typeset different margin texts with different characteristics like bodyfont, line spacing and offset.

Isn't
this
cute?

```
\margintext{Isn't}
\margintext{this}
\margintext{cute?}
```

In the setup we see an optional argument. The number is determined by the order of definition.

```
\setupinmargin[1][align=right, line=1,style=slanted]
\setupinmargin[2][align=middle,line=2,style=boldslanted]
\setupinmargin[3][align=left,  line=3,style=bold]
```

This means that the second margintext in a row will start on line 2, and be typeset in a bold slanted font. One can explicitly force a margintext to go some place, by saying for instance:

```
\margintext[2]{this is the second one}
```

4.9 Subscript and superscript

There are three commands to create superscript and subscript outside the math mode:

```
\high{...}
...     text
```

```
\low{...}
...     text
```

```
\lohi{.1.}{.2.}
.1.      text
.2.      text
```

The next example illustrates the use of these commands:

```
You can walk on \high {high} heels or \low {low} heels but your height
is still the same.
```

This results in:

You can walk on $^{\text{high}}$ heels or $_{\text{low}}$ heels but your height is still the same.

These commands relate to the ˆ and _ in math mode. In case of larger fontsizes like `\tfc`, the ˆ and _ will not create the desired output. Compare the examples below:

```
test\high{test} test test$ˆ{\rm test}$ test
{\bf test\high{test} test test$ˆ{\bf test}$ test}
{\tfb test\high{test} test test$ˆ{\tfb test}$ test}
```

This becomes:

test$^{\text{test}}$ test test$^{\text{test}}$ test
test$^{\textbf{test}}$ test test$^{\textbf{test}}$ test
test$^{\text{test}}$ test test$^{\text{test}}$ test

4.10 Columns

The TEX programmer knows that it is not easy to put text in columns. Gratefully a CONTEXT user is not bothered with the implementation of extensive macros.

You can typeset text in columns. Most commands can be used in a normal way without any problems. The floating object like tables or figures are somewhat limited. This is caused by the fact that TEX has limited capabilities for typesetting columns. For insiders: columns are produced with the primitives: `\output` and `\vsplit`.

The number of columns is unlimited, however TEXs memory can only handle upto about twenty to thirty or fourty columns.

The number of columns and the type setting of a vertical line as a column separator is set up by:

```
\setupcolumns[..,..=..,..]

n            number
rule         on off
height       dimension
tolerance    verystrict strict tolerant verytolerant stretch
distance     dimension
balance      yes no
align        yes no text
blank        fixed halfline line flexible big medium small
option       background
direction    left right
..=..        see p 264: \setupframed
```

The `n` indicates the number of columns. The column text is enclosed by:

```
\startcolumns[..,..=..,..] ... \stopcolumns

..=..    see p 79: \setupcolumns
```

The local setup of columns can be added directly after this command. A new column is forced by:

```
\column
```

The text below is typeset in two columns with a `verytolerant` alignment.

```
\startcolumns[rule=on,n=2,tolerance=verytolerant]
Thus, I came to the conclusion that the designer of a new
system must not only be the implementer and first
.
.
\bf D.E. Knuth
\stopcolumns
```

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual. The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

D.E. Knuth

This example makes it painfully obvious that spacing between lines is not on forehand equal. By default the line spacing in this document is `big`, which equals .75×`\lineheight`. Furthermore, the allowable stretch in line spacing makes vertical alignment practically impossible.

For this reason the default line spacing is equal to the lineskip and stretching is not allowed. When a switch in fontsize is desirable you should do so before starting the column mechanism. Font switches within columns will have a poor result. The next example shows a line spacing equal to the lineskip.

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

D.E. Knuth

This effect is reached by the (default) setup:

```
\setupcolumns[blank={fixed,line}]
```

In section 3.4 typesetting on a grid is explained. This mechanism works quite well within columns.

TEX is not an easy to learn typesetting system or program. The problem is that “knowing everything is possible” leads to “wanting everything that is possible”. However using CONTEXT or TEX takes considerable learning time. And it is not feasible to explain every single detail in this manual. Therefore “doing” is the answer.

This text shows that one can do some tricks with columns. The frame is created by:

```
\def\FramedColumn#1{\ruledhbox{\box#1}}
```

```
\setupcolumns[command=\FramedColumn]
```

A less senseless display is:

```
\def\FramedColumn#1%
  {\hbox to \hsize
     {\ifodd\currentcolumn\unhbox\hss#1\else\unhbox#1\hss\fi}}
```

This time the columns will look like:

TEX is not an easy to learn typesetting system or program. The problem is that “knowing everything is possible” leads to “wanting everything that is possible”. However using CONTEXT or TEX takes considerable learning time. And it is not feasible to explain every single detail in this manual. Therefore “doing” is the answer.

A column can be manipulated as a whole. For example to create a background:

```
\setupfootnotes
  [location=columns,
   background=color,
   backgroundcolor=white]
```

```
\setuplayout
  [grid=yes]
```

This time the column will be typeset on a grid:

TEX is not an easy to learn typesetting system or program. The problem is that “knowing everything is possible” leads to “wanting everything that is possible”. However using CONTEXT or TEX takes considerable learning time. And it is not feasible to explain every single detail in this manual. Therefore “doing” is the answer.

4.11 Paragraphs in columns

In some cases you want to typeset a paragraph in columns. For example in a definition where you have a first column containing meaningful text and a second column containing meaningful text. In these cases you can use:

```
\defineparagraphs[...][..,..=..,..]
...          name
n            number
rule         on off
height       fit dimension
before       command
after        command
inner        command
distance     dimension
tolerance    verystrict strict tolerant verytolerant stretch
align        left right middle
```

This command defines a column layout that is recalled by its name.

```
\startparagraph ... \stopparagraph
```

The layout can be set up by:

```
\setupparagraphs[.1.][.2.][..,..=..,..]
.1.          name
.2.          number each
style        normal bold slanted boldslanted type cap small... command
width        dimension
height       dimension
align        left right middle width breedte
tolerance    verystrict strict tolerant verytolerant stretch
distance     dimension
before       command
after        command
inner        command
command      command
rule         on off
```

The width of non–specified columns is determined automatically. Distance relates to horizontal white space in front of a column. The next column is specified by:

```
\paragraph
```

We show a simple example of the use of paragraphs in columns.

4 Paragraphs in columns

```
\defineparagraphs[TwoColumns][n=2]
\setupparagraphs[TwoColumns][1][width=5cm]

\startTwoColumns
  This is the top left corner.
\TwoColumns
  In graphic environments the top right corner is also called the upper
  right corner.
\stopTwoColumns

\startTwoColumns
  In a similar way, the bottom left corner is called the lower left corner.
\TwoColumns
  Which leaves the bottom right corner, that is also known as lower right
  corner. Now what is the alternative name for the top left corner?
\stopTwoColumns
```

Here the `\TwoColumns` separates the columns. With a default setup this results in:

This is the top left corner.	In graphic environments the top right corner is also called the upper right corner.
In a similar way, the bottom left corner is called the lower left corner.	Which leaves the bottom right corner, that is also known as lower right corner. Now what is the alternative name for the top left corner?

We also could have used `\nextTwoColumns` instead of `\TwoColumns`. Sometimes this is more readable in your ASCII text. An alternative specification is:

```
\TwoColumns first text \\ second text \\
```

You can add a command to the keywords `bottom` and `top`. These commands will be executed before or after the text. For example a column can be forced down by `[top=\vfill]`.

This is the right place to show a more complex example. The use of `paragraphs` is preferred over the use of columns because the text is kept together. If we want to score an item on two dimensions we need three columns:

```
\defineparagraphs [CombinedItem]     [n=3,rule=on]
\setupparagraphs  [CombinedItem] [2] [width=3em]
\setupparagraphs  [CombinedItem] [3] [width=7em]
```

The item itself is defined with `\defineenumeration` (see section ??):

```
\defineenumeration
  [SomeItem]
  [location=left,text=,width=3em,before=,after=]
```

The scoring is done on a scale that is typeset as an itemization (see section ??). An item might look like this in ASCII:

```
\startCombinedItem
  \startSomeItem
    The student is able to write a detailed planning for the
    design and construction of a water purification plant.
  \stopSomeItem
\nextCombinedItem
  \startitemize[5,packed]
    \item yes \item no
  \stopitemize
\nextCombinedItem
  \startitemize[5,packed]
    \item self study \item class room \item simulation
  \stopitemize
\stopCombinedItem
```

And will result in:

1	The student is able to write a detailed planning for the design and construction of a water purification plant.	∘ yes ∘ no	∘ self study ∘ class room ∘ simulation

When the scoring scales are identical over all items we can use macros:

```
\def\firstscale%
  {\startitemize[5,packed]
     \item yes \item no
   \stopitemize}

\def\secondscale%
  {\startitemize[5,packed]
   \item self study \item class room \item simulation
   \stopitemize}

\startCombinedItem
  \startSomeItem
```

```
    The student is able to write a detailed planning for the
    design and construction of a water purification plant.
  \stopSomeItem
\nextCombinedItem
  \firstscale
\nextCombinedItem
  \secondscale
\stopCombinedItem
```

Or even more sophisticated:

```
\def\startItem%
  {\startCombinedItem
   \startSomeItem}
```

```
\def\stopItem%
  {\stopSomeItem
   \nextCombinedItem \firstscale
   \nextCombinedItem \secondscale
   \stopCombinedItem}
```

```
\startItem
  The student is able to write a detailed planning for the
  design and construction of a water purification plant.
\stopItem
```

A definition like the one above can be very surprising. The commands in such a definition can interfere and result in undesirable output. We think of `\vtop`'s that align on the baseline and `\vbox` s that align under the baseline. Another example with framed texts show that ConTeXt takes care of most of the problems.

left	middle	right

4.12 Tabulate

In a later chapter we will go into detail on typesetting tables. Consider this paragraph to be an appetizer. We use the term tabulate when a table is part of the running text. A simple tabulation looks like this:

```
\starttabulate[|l|p|]
\NC question \NC Sometimes it is surprising to notice that writers,
independently of each other, explore the same theme along similar lines.
Three of the four books mentioned here fall into this category. Which
books do not belong in this list? \NC \NR
\stoptabulate
```

```
\starttabulate[|l|l|l|]
\NC A. \NC This Perfect Day           \NC Ira Levin           \NC \NR
\NC B. \NC Opstaan op Zaterdag        \NC Jan Gerhart Toonder \NC \NR
\NC C. \NC Tot waar zal ik je brengen \NC Anton Koolhaas      \NC \NR
\NC D. \NC The City And The Stars     \NC Arthur Clarke       \NC \NR
\stoptabulate
```

This results in:

question	Sometimes it is surprising to notice that writers, independently of each other, explore the same theme along similar lines. Three of the four books mentioned here fall into this category. Which books do not belong in this list?

A.	This Perfect Day	Ira Levin
B.	Opstaan op Zaterdag	Jan Gerhart Toonder
C.	Tot waar zal ik je brengen	Anton Koolhaas
D.	The City And The Stars	Arthur Clarke

With `\NC` we go to the next column and with `\NR` to the next row. Definitions like `[|l|p|]` and `[|l|l|l|]` are called a template. The set ups are similar to those of `\starttable` (see in ??).

The default template looks like this: `[|l|p|]`. The second column is typeset as a normal paragraph and with a width that is calculated automatically by TEX.

```
\starttabulate
\NC d: \NC avond, afond, avend, afend \NC \NR
\NC t: \NC avont, afont, avent, afent \NC \NR
\stoptabulate
```

This quotation from “Spellingsverandering van zin naar onzin” by G.C. Molewijk (1992) will look like this:[4]

[4] For the non–dutch readers: this book “Change of spelling, from sense to nonsense” is one of the most humorous books on the developments in a language one can imagine. If you ever come to studying dutch, you should give this book a try.

d: avond, afond, avend, afend
t: avont, afont, avent, afent

4.13 Alignment

Horizontal and vertical alignment is set up by:

```
\setupalign[...]
...      width left right middle inner outer wide broad height bottom line reset
```

The keys `left`, `middle` and `right`, `inner` and `outer` apply to horizontal alignment and `bottom`, `height` and `line` to vertical alignment.

The key `right` results in the text being typeset ragged right. The keyword `broad` can be combined with `left`, `middle` and `right` which results in somewhat more rough alignments.

The option `line` lets the last line touch the bottom of the page while `height` aligns the baseline to the bottom.

Individual lines can be aligned with the commands:

```
\leftaligned{...}
...      text
```

```
\midaligned{...}
...      text
```

```
\rightaligned{...}
...      text
```

alignment over a number of lines is done by:

```
\startalignment ... \stopalignment[...]
..=..    see p 87: \setupalign
```

The text below shows a number of examples of horizontal alignment.

The Brittish stubbornly stick to
driving at the left side of the road.

This can be considered a form conservatism,
or alternatively phrased: right–wing thinking.

However, a political drive–in–the–middle
compromise would definitely lead to accidents.

We done this with:

```
\leftaligned{The Brittish stubbornly stick to}
\leftaligned{driving at the left side of the road.}
\blank[medium]
\rightaligned{This can be considered a form conservatism,}
\rightaligned{or alternatively phrased: right||wing thinking.}
\blank[medium]
\midaligned{However, a political drive||in||the||middle}
\midaligned{compromise would definitely lead to accidents.}
```

The last words of a paragraph can be placed on the right hand side by the command `\wordright`, **so with:**

```
\wordright{...}
...      text
```

When typesetting a paragraph, TEX tries several alternatives and decides which one to choose based on a system, of penalties. Normally TEX is very strict, but we can instruct TEX to be a bit more tolerant. This means that, instead of letting problematic situations remain unsolved —i.e. let words that cannot be hyphenated stick into the margin— TEX will add a bit more stretch and apply different penalties for successive hyphens.

Alignment can be set up by:

```
\setuptolerance[..,...,..]
...      horizontal vertical stretch verystrict strict tolerant verytolerant
```

By default we use `[horizontal,verystrict]` for horizontal alignment and `[vertical,strict]` for vertical alignment.[5] A last resort is provided by the keyword `stretch`,

[5] If you want a real ugly result, you should set the TEX variable `\pretolerance` to 10.000. It is up to you.

which in unsolvable situations will stretch spaces, extending the ugliness even further.

In double sided typesetting, alignment can be coupled to the left or right pages.

```
\startalignment[inner]
\quotation {Out of nowhere} is a rather normal way of saying that it is
not clear where something originates. It is typically a phrase that has
no counterpart, in the sense that nobody would comprehend the remark
\quotation {Into somewhere}.
\stopalignment
```

```
\startalignment[outer]
\quotation {Out of bounds} is a similar quote. There is no counterpart
\quotation {In of bounds}. Both examples demonstrate that in(ner) and
out(er) are not always counterparts.
\stopalignment
```

Results of the commands above depend on the location of the page (left of right). The commands lead to:

“Out of nowhere” is a rather normal way of saying that it is not clear where something originates. It is typically a phrase that has no counterpart, in the sense that nobody would comprehend the remark “Into somewhere”.

“Out of bounds” is a similar quote. There is no counterpart “In of bounds”.
Both examples demonstrate that in(ner) and out(er) are not always counterparts.

4.14 New lines

A new line is forced by:[6]

```
\crlf
```

If you want to have lines show up the way you typed them in your source file you can use:

```
\startlines ... \stoplines
```

Default indenting is off. You can set up lines by:

[6] In titles, headers and margin texts \\ is available for introducing a new line.

```
\setuplines[..,..=..,..]

before       command
after        command
inbetween    command
indenting    yes no even odd
```

If we set up `indenting=odd` for example we will obtain:

Come on, he said, give me a while,
    and I will typeset you this text
with rivers like the river Nile

This was typed in the source file as:

```
\setupindenting[medium]
\setuplines[indenting=even]
\startlines
Come on, he said, give me a while,
and I will typeset you this text
with rivers like the river Nile
\stoplines
```

Lines can be numbered with:

```
\startlinenumbering[...] ... \stoplinenumbering

...     continue
```

A simple example of numbered lines might look like this:

```
\startlinenumbering
There is of course no problem with trying to prevent illegal copying of
\cap {cd}'s and records. However, why should artists benefit from these
measures, who themselves have no problems with copying themes, lyrics
and melodies?
\stoplinenumbering
```

this becomes:

There is of course no problem with trying to prevent illegal copying of CD's and records. However, why should artists benefit from these measures, who themselves have no problems with copying themes, lyrics and melodies?

We can influence line numbering by:

```
\setuplinenumbering[..,..=..,..]

conversion     numbers characters Characters romannumerals Romannumerals text
start          number
step           number
width          dimension
location       intext inmargin
style          normal bold slanted boldslanted type cap small... command
prefix         text
referencing    on off
```

With the variable `conversion` you set up the type of numbering. You may even use your own character, for example an em–dash (keyed in as `---`). In that case this character is set in front of each line.

In chapter 9.5 we will explain how we can refer to a linenumber. The parameters `prefix` and `referencing` can be used to unfluence that proces.

In the example below we use the following setup:

`\setuplinenumbering[conversion=numbers,step=2,location=intext]`

and:

`\setuplinenumbering[conversion=characters,step=1,location=intext]`

```
    a macro is a piece of text          a   but when fed to TEX the program
2   random at first sight               b   you will be surprised
    a bunch of stupid tokens that       c   thanks to macros your text too
4   looks less that awful right         d   will look quite organized
```

You can also mark lines in order to refer to specific line numbers. This will be shown in in chapter 9.5.

4.15 New page

In some instances it is up to you to force, prevent or encourage a new page.

```
\page[..,...,..]

...     yes markup no preference bigpreference left right disable last quadruple even odd
        blank empty reset
```

The possible set ups are explained in table 4.3. If no setup is used `\page` will result in a new page.

setup	**result**
`yes`	force a new page
`makeup`	the same, without fill
`no`	when possible, avoid page break
`preference`	when possible, force page break
`bigpreference`	when possible, force page break, try harder
`left`	force a left page
`right`	force a right page
`disable`	ignore the next `\page` command
`last`	add last page(s)
`quadruple`	add pages until quadruple number of pages
`even`	go to the next even page
`odd`	go to the next odd page
`blank`	insert a completely blank page
`empty`	insert an empty page (with headers etc.)
`reset`	reset the disable command

Table 4.3 Setups of `\page`.

The setups `last` and `quadruple` can be used in double sided (reduced) typesetting. The first setup up will add pages until an even number is obtained, the second set up will add pages until the next quadruple is reached. When you want to overrule the automatic page numbering you type the pagenumber yourself:

```
\page[25]
```

You can also use a relative number like `[+4]`. You can use this feature when you want to be on the safe side and if you don't know at what page you are.

While generating empty pages you have to take doublesidedness into account, for example:

```
\page[right,empty,right]
```

4.16 Pagenumbers

At any location in the text the pagenumber can be set up with the command:

```
\setuppagenumbering[..,..=..,..]

alternative        singlesided doublesided
location           header footer left right middle margin marginedge inleft inright
conversion         numbers characters Characters romannumerals Romannumerals
style              normal bold slanted boldslanted type cap small... command
left               text
right              text
way                bytext bysection bypart
text               text
numberseparator    text
textseparator      text
sectionnumber      yes no
separator          text
state              start stop
command            \command#1
```

The pagenumber position on the page is defined by:

```
\setuppagenumbering[..,..=..,..]

alternative        singlesided doublesided
location           header footer left right middle margin marginedge inleft inright
conversion         numbers characters Characters romannumerals Romannumerals
style              normal bold slanted boldslanted type cap small... command
left               text
right              text
way                bytext bysection bypart
text               text
numberseparator    text
textseparator      text
sectionnumber      yes no
separator          text
state              start stop
command            \command#1
```

The position varies with the nature of the document. With `conversion` we state the way we want to display the number. With `location` we define pagenumber positions like the bottom or top, left or right side or in the margin. You can use combinations of these options. For example:

```
\setuppagenumbering[location={header,inmargin}]
```

alternative=singlesided	alternative=doublesided
left, right	marginedge
middle	middle
margin	margin

Table 4.4 setups to `\setuppagenumbering`.

Another alternative is `{singlesided,doublesided}`. In this case headers and footers will be mirrored in a double-sided document. The backspace is not mirrored (see figure 4.1).

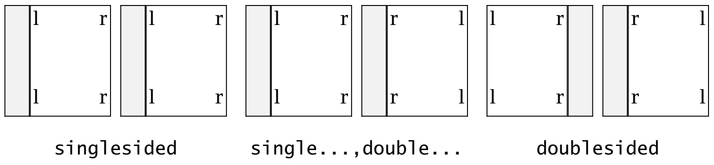


Figure 4.1 Three ways to mirror.

You can assign text to the parameters `left` and `right`. These texts will encloses the pagenumber:

```
\setuppagenumbering[conversion=romannumerals,left={--~},right={~--}]
```

This will lead to: – viii –. With `style` you define the font and with `state` pagenumbering is switched on and off.

Numbering can become very fancy when you use `command` to execute an operation. This command has an argument and will be executed every time a pagenumber is placed. A framed pagenumber can be obtained by:

```
\setuppagenumbering[command=\inframed]
```

or partially framed by:

```
\def\mypagenumber#1%
  {\inframed[frame=off,leftframe=on,rightframe=on]{#1}}
```

```
\setuppagenumbering[command=\mypagenumber]
```

In this we use `\inframed` instead of `\framed`, because the pagenumber must align with the texts of the headers and footers.

With `textseparator` you can define a separator between the section and pagenumber. Default this is a –. When the pagenumber is to appear at the margin the `numberseparator` is placed between the number and the footer text. Default this is a space with a width of 1em.

In interactive documents subpagenumbering is frequently used for hyperlinking. When every new section is started on a new page the footer text can be set up with:

```
\setupsubpagenumber
  [way=byparagraph]
\setupfootertexts
  [screen {\subpagenumber} of {\numberofsubpages}] []
```

The setup is done with:

```
\setupsubpagenumber[..,..=..,..]

way      bytext bysection bypart
state    start stop none
```

and the numbers themselves can be recalled by `\subpagenumber` and `\numberofsubpages`. These numbers are only reliable in headers and footers. In the case of interactive documents a more abstract definition can be used:

```
\setupfootertexts[][{\interactionbar[alternative=d]}]
```

In this case one can jump to the previous and following subpages. The subnumbering can be reset with `[reset]`.

In a similar fashion one has access to the page number and the total number of pages: `\pagenumber` and `\totalnumberofpages`.

4.17 Headers and footers

Text in the header and footer are set up with the commands:

```
\setupheadertexts[.1.][.2.][.3.]
.1.     text margin edge
.2.     text section date mark pagenumber
.3.     text section date mark pagenumber
```

```
\setupfootertexts[.1.][.2.][.3.]
.1.     text margin edge
.2.     text section date mark pagenumber
.3.     text section date mark pagenumber
```

A great number of arguments can be added. When the first argument is left out it is taken for granted that the footer and header should be place under or over the pagebody (`text`). The edge is located at the left side of the margin and is only used in interactive documents where a extended pagebody is needed.

The key `date` generates a date and `pagenumber` generates the pagenumber. Part, chapter and section titles can be summoned to appear in the header– and footer text by `part`, `chapter`, `paragraph` etc. By default the mark mechanism is active. Sectionnumbers can also be recalled: `chapternumber` etc.

Setting the `state` is done for the whole header, so one should use the one–argument version:

```
\setupheader[state=high]
```

Those who want more variations in headers and footers can use four instead of two arguments. Four arguments have only effect in double–sided documents.

```
\setupfootertexts
  [even left][even right]
  [odd left][odd right]
```

So there are different combinations of arguments possible:

```
\setupheadertexts
\setupheadertexts[mid text]
\setupheadertexts[left text][right text]
\setupheadertexts[left text][right text][left .][right .]
\setupheadertexts[location][left text][right text]
\setupheadertexts[location][left text][right text][left .][right .]
```

Instead of text, one can specify keywords like `chapter`, `date` or `pagenumber`. When the pagenumber is positioned in this way, one should also say:

```
\setuppagenumbering[location=]
```

The current setups of the headers and footers are cleared when no values are stated in `\setupfootertexts`. Problems can be expected when you use `[ ]` in your setup. These have to be enclosed in curly brackets:

```
\setupfootertexts[chapter][{\currentdate[month,year]}]
```

The type setting of head– and foot texts can be influenced by:

```
\setupheader[...][..,..=..,..]

...           text margin edge
state         normal stop start empty high none nomarking name
strut         yes no
style         normal bold slanted boldslanted type cap small... command
leftstyle     normal bold slanted boldslanted type cap small... command
rightstyle    normal bold slanted boldslanted type cap small... command
leftwidth     dimension
rightwidth    dimension
before        command
after         command
```

and

```
\setupfooter[...][..,..=..,..]

...       see p 97: \setupheader
..=..     see p 97: \setupheader
```

As with `\setup...texts` the first argument is optional. The keys `state`, `before` and `after` work on all parts of the pagebody, on the main text, the margins and edges.

When `...width` is set up the text is clipped at the given width. The key `strut` is important when footers or headers contain other objects than text. When `strut` is set to `no`, the object is not corrected for linedepth. You could use the command `\showstruts` to get some information on this phenomena.

The setups with `state` are explained in table 4.5. You should bear in mind that page numbering will always continue whether or not the pagenumbers are placed.

setup	result
`normal`	visible
`none`	invisible, no whitespace
`empty`	one page invisble, whitespace
`high`	one page visible, no whitespace
`start`	visible
`nomarking`	leave out marks
`stop`	invisible, whitespace

Table 4.5 Setups with `\setupheader` and `\setupfooter`.

When setups are done between `\start` and `\stop` they will only work locally. This means that the setups are reset after `stop`. Headers and footers may appear even while you think new ones should appear. This is due to the way TEX determines valid breakpoints. One can never be certain when such an automatic break will occur. The solution is to force a new page by `\page` before `\stop`.

Headers and footers can be switched off on a page by means of:

```
\noheaderandfooterlines
```

Next to head– and footertexts there are also over– and bottomtexts. These are setup in a similar way:

```
\setuptoptexts[.1.][.2.][.3.]
.1.     text margin edge
.2.     text section date mark pagenumber
.3.     text section date mark pagenumber
```

```
\setuptexttexts[.1.][.2.][.3.]
.1.     text margin edge
.2.     text section date mark pagenumber
.3.     text section date mark pagenumber
```

```
\setupbottomtexts[.1.][.2.][.3.]
.1.      text margin edge
.2.      text section date mark pagenumber
.3.      text section date mark pagenumber
```

```
\setuptop[...][..,..=..,..]
...      see p 97: \setupheader
..=..    see p 97: \setupheader
```

```
\setuptext[...][..,..=..,..]
...      see p 97: \setupheader
..=..    see p 97: \setupheader
```

```
\setupbottom[...][..,..=..,..]
...      see p 97: \setupheader
..=..    see p 97: \setupheader
```

```
\notopandbottomlines
```

When the height of an area equals zero, no text is placed. By default the top and bottom area have zero height, so setting their text areas without setting the height has no effect.

At the instance of a new part or chapter we can deal in a different way with the headers and footers. Suppose that a default setup looks like this:

```
\setupheadertexts[pagenumber]
\setupfootertexts[chapter][paragraph]
```

At the first page of new chapters this may look not too good. Therefore we could state:

```
\setuphead[chapter][header=empty,footer=empty]
```

However if we use it in this way we loose the pagenumber. A more adequate solution is:

```
\definetext[chapter][footer][pagenumber]
```

with:

```
\setuphead[chapter][header=high,footer=chapter,page=right]
```

we obtain the desired effect. The pagenumber appears in the foot and the header disappears completely. These kind of commands are essential when you don't want to define all kinds of setups locally in a text, for example before every new chapter. This mechanism only works when going to a new page enabled.

```
\definetext[.1.][.2.][.3.][.4.][.5.]

.1.     name
.2.     header footer
.3.     text
.4.     text
.5.     text
```

4.18 Footnotes

In some texts you can't do without footnotes. The footnote marker is placed in the text and the note itself is typeset at another location in the text, usually at the bottom of the page. Most often at the bottom of the page.

```
\footnote[ref]{...}

...     text
```

A footnote number or –symbol is recalled with:

```
\note[ref]
```

An example of footnotes is given below.

```
The first compositions of the American composer Steve Reich will probably
only appreciated by the most \quote {purist} among those who like
minimal||music \footnote {A decent minimal is not so much characterized by
a minimal use of musical instruments, but more by subtle shifts in
polyphonic rhythms.}, his later works, like \quote {The Desert Music}, are
compositions for full orchestra, where the orchestra is extended with a for
Reich characteristic rhythm section \footnote {In most cases this section
consists of pianos, marimbas and xylophones.} and choir. Together
```

```
with John Adams, \footnote {His \quote {Fearful Symmetries} is a perfect mix
of classic, jazz, swing and pop music.} Reich can be considered one of
today’s leading composers. It is, however, a pity that they can only be seen
\footnote {The nice thing about compositions like \quote {Drumming} and
\quote {Sextet} is de fact that \quotation {what the ear hears} differs
from what the \quotation {eye sees happening}.} and heard at the smaller
broad companies, like the \cap {VPRO}. \footnote{A non commercial Dutch
broadcast company.} \footnote {Sometimes also at other companies, because
somehow this kind of music is quite suited for impressive and|/|or
melodramatic documentaries.}
```

Undesired spaces are ignored. Spacing between two footnote numbers or symbols is taken care of. The result looks like this:

The first compositions of the American composer Steve Reich will probably only appreciated by the most ‘purist’ among those who like minimal–music[7], his later works, like ‘The Desert Music’, are compositions for full orchestra, where the orchestra is extended with a for Reich characteristic rhythm section[8] and choir. Together with John Adams,[9] Reich can be considered one of today’s leading composers. It is, however, a pity that they can only be seen[10] and heard at the smaller broad companies, like the VPRO.[11] [12]

The type setting of the footnote can be setup with the command below that is defined in the setup area of your document.

7 A decent minimal is not so much characterized by a minimal use of musical instruments, but more by subtle shifts in polyphonic rhythms.
8 In most cases this section consists of pianos, marimbas and xylophones.
9 His ‘Fearful Symmetries’ is a perfect mix of classic, jazz, swing and pop music.
10 The nice thing about compositions like ‘Drumming’ and ‘Sextet’ is de fact that “what the ear hears” differs from what the “eye sees happening”.
11 A non commercial Dutch broadcast company.
12 Sometimes also at other companies, because somehow this kind of music is quite suited for impressive and/or melodramatic documentaries.

```
\setupfootnotes[..,..=..,..]
```

conversion	<u>numbers</u> characters Characters romannumerals Romannumerals
way	<u>bytext</u> by*section*
location	<u>page</u> text columns high
rule	on <u>off</u>
before	*command*
after	*command*
width	*dimension*
height	*dimension*
bodyfont	5pt ... 12pt <u>small</u> big
style	normal bold slanted boldslanted type cap small... *command*
distance	*dimension*
columndistance	*dimension*
margindistance	*dimension*
n	*number*
numbercommand	\command#1
..=..	see p 258: \framed

By default footnotes are placed at the bottom of a page. When using columns you can set `location` to `columns` so that the footnotes appear in the last column.

We can frame footnotes, place them in columns and decouple them from a page. The meaning of this last option is explained in an example.

```
\startlocalfootnotes[n=0]
  \placetable
    {A (latin) table.}
    \placelegend
      {\starttable[|l|r|]
       \HL
       \VL Nota \footnote {Bene} \VL Bene \footnote {Nota} \VL\FR
       \VL Bene \footnote {Nota} \VL Nota \footnote {Bene} \VL\LR
       \HL
       \stoptable}
      {\placelocalfootnotes}
\stoplocalfootnotes
```

The table enables the float placement mechanism, so we don't know on which page the table nor the footnotes will appear. So the footnotes are coupled to the table by using local footnotes.

Nota[1]	Bene[2]
Bene[3]	Nota[4]

[1]Bene [2]Nota [3]Nota [4]Bene

Table 4.6 A (latin) table.

```
\startlocalfootnotes ... \stoplocalfootnotes
..=..    see p 102: \setupfootnotes
```

```
\placelocalfootnotes[..,..=..,..]
..=..    see p 102: \setupfootnotes
```

Footnotes can be placed at the end of a chapter or a document. The key `location` is set at `text` and we use the following command to place the footnotes:

```
\placefootnotes[..,..=..,..]
..=..    see p 102: \setupfootnotes
```

When `n` is set at 2, you can display the footnotes in columns. This should be done at an early stage because TEX is using the dimensions of the footnotes to determine the page break. More information can be found in the source code of the CONTEXT module: `core-not.tex`.

The next example demonstrates that footnote numbers can be replaced by footnote symbols. In this example `conversion` is set at `set 3`.

note: use footnotes sparingly*
note: be brief**
note: no notes are even better***

Default the key `numbercommand` is set `\high`, but other setups are allowed. You can also work with:

* During the development of CONTEXT the footnote mechanism was one of the first real challenges. And I'm challenged still since I just encountered documents with footnotes within footnotes.

** Why? See note*.

*** QED.

```
\setupfootnotedefinition[..,..=..,..]
..=..    see p 225: \definedescription
```

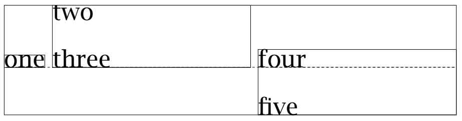
to define the exact way of how to display the footnotes, because the standard definition mechanism is used (see section ??).

4.19 Aligned boxes

TEX is basically aware of two kind of boxes: `\hbox` and `\vbox`. A horizontal `\hbox` can be considered a line, a `\vbox` a paragraph. There are two types of vertical boxes: a `\vbox` aligns on the baseline of the last line, while a `\vtop` aligns on the first line.

```
\hbox{\hbox{one} \vbox{two\par three} \vtop{four\par five}}
```

When we make the frames visible —in this case we said `\showboxes` in advance— the example above becomes:



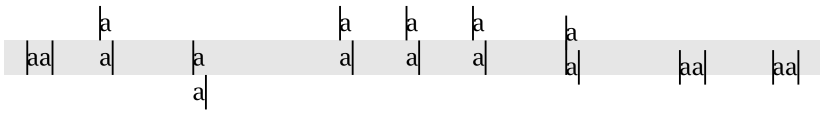
In addition CONTEXT provides a lot of alternative boxes, like: `\cbox`, `\lbox` and `\rbox`. These commands can be used while defining your own macros, but will seldom appear in the running text. Like in `\hbox` and `\vbox` the dimension of the width can be added.

```
\cbox{... text ...}
\lbox to 4cm{... text ...}
```

The reader is invited to experiment with these commands. A new line is forced with `\\`.

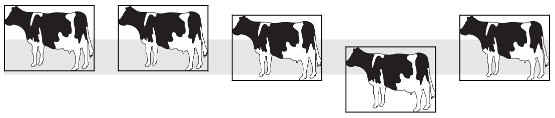
For some very dedicated purposes there is `\sbox`. This command is used to give a box the height of a strut. You may forget this command.

To another category of boxes belong `\tbox` and `\bbox`. Both are used within tables. Look at the example below that illustrates their use.



```
\hbox  \vbox  \vtop  \lbox  \cbox  \rbox  \sbox  \tbox  \bbox
```

The `\tbox` and `\bbox` are also used in figures.



`\hbox` `\vbox` `\sbox` `\tbox` `\bbox`

In ConTeXt a complete repertoire of macros is available that relies on boxes. For example we can add cutmarks to a box:

```
\setbox0=\vbox{The Final Cut\par --- \em Pink Floyd}
\makecutbox0 \box0
```

Be aware of the fact that such marks lie outside the boxes.

The Final Cut

— *Pink Floyd*

We can visualize boxes by using `\ruledhbox`, `\ruledvbox` and `\ruledvtop` instead of `\hbox`, `\vbox` and `\vtop`. With `\showmakeup` we can visualise everything automatically and we can get some insight on the features of ConTeXt and TeX.

The next example shows that we can use TeX for more than only the straight forward typesetting. However, to be able to do this, one should have some insight in the manipulation of boxes. We use buffers to enhance comprehensibility.

```
\startbuffer[water]
Drink geen water \crlf direct uit de kraan! \blank

\start
  \tfx \setupinterlinespace Het drinkwater is tijdelijk niet betrouwbaar.
  Kook het water voor consumptie ten minste 2~minuten. Zodra het water
  weer betrouwbaar is, krijgt u bericht. \par
\stop

\blank[2*big]

\language[en] Do not drink water \crlf directly from the tap! \blank

\start
  \tfx \setupinterlinespace The water is temporarily unfit for drinking.
  Boil the water during at least 2~minutes before consumption. As soon
  as the water is reliable again, you will be notified. \par
```

```
\stop
\stopbuffer
```

This text is typeset in a framed box. We use two temporary boxes. The first determines the height of the second one. Instead of `\tfx\setupinterlinespace` you could use `\switchtobodyfont` to switch to a narrower bodyfont. (`[small]`). The `\par` is essential!

```
\framed[offset=\bodyfontsize]
  {\setbox0=\vbox
     {\hsize 16em\switchtobodyfont[ss]\getbuffer[water]}
   \setbox2=\vbox to \ht0
     {\vfill\externalfigure[vew1091a][width=5cm]\vfill}
   \hskip1em\box2\hskip1em\box0\hskip1em}
```

The result —an example of a drinking water warning— is shown below.

Drink geen water
direct uit de kraan!

Het drinkwater is tijdelijk niet betrouwbaar. Kook het water voor consumptie ten minste 2 minuten. Zodra het water weer betrouwbaar is, krijgt u bericht.

Do not drink water
directly from the tap!

The water is temporarily unfit for drinking. Boil the water during at least 2 minutes before consumption. As soon as the water is reliable again, you will be notified.

4.20 Makeup

A document may have a titlepage, a colofon and some pages that are not directly related to the main part of the document. Mostly these pages are not numbered and can do without headers and footers. Because their layout needs extra attention we prefer the word makeup for defining their specific layout.

The commands \startstandardmakeup and \stopstandardmakeup exclude text from the standard pagebody and its layout. Below a simple example is given. You will notice commands like \vfill, \blank, \tf and even \crlf and \vskip.

```
\startstandardmakeup
  \tfd Jobs around the house \blank[2*big]
  \tfb Part 1: Gas, water and  electricity \vfill
  \tfb J. Hagen \crlf A.F. Otten \blank
  \tfb Hasselt \crlf \currentdate[month,year]
\stopstandardmakeup
```

In double–sided documents an empty page is generated that functions as the backside of the title page. However sometimes this backside should also be typeset.

```
\startstandardmakeup[doublesided=no]
... the front
\stopstandardmakeup
\startstandardmakeup[page=no]
... the back
\stopstandardmakeup
```

Because double–sided typesetting is turned off, a backside page is not generated. And because the key page is no the next page does not get the layout of a right hand side page (this would be default).

With the command \showframe frames can be made visible (temporarily) around the made up text. This is very convenient during the typesetting of separate pages.

Next to the command \startstandardmakeup one can define his own layout with different dimensions by means of:

```
\definemakeup[...][..,..=..,..]

...      name
..=..    see p 108: \setupmakeup
```

```
\setupmakeup[...][..,..=..,..]
...            name
width          dimension
height         dimension
voffset        dimension
hoffset        dimension
page           left yes right
commands       command
doublesided    yes no empty
headerstate    normal stop start empty none nomarking
footerstate    normal stop start empty none nomarking
textstate      normal stop start empty none nomarking
topstate       stop start
bottomstate    stop start
```

```
\startnamemakeup ... \stopname
```

The first command generates a `\start...stop`–pair between which the new typesetting commands can be typed. Bij default the result of this new layout is typeset on an empty page. The new layout is marked with *name*, for selection at a later stage (see section ??).

The commands that are provided after the key `commands` are executed immediately when a new layout is called. In this local layouts can be defined.

Typography

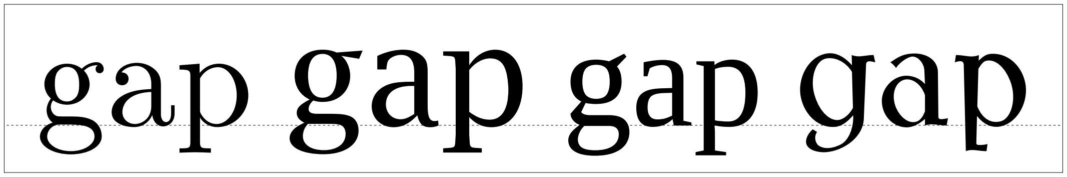
5

5.1 Introduction

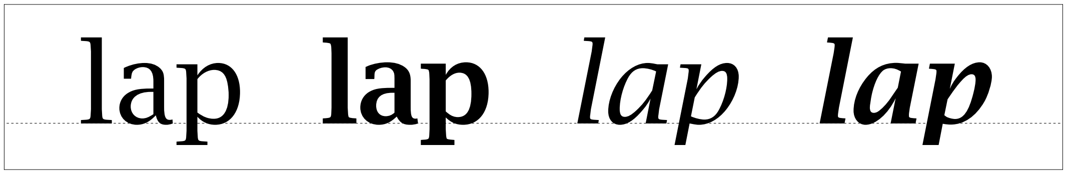
Through the millennia we have developed and adapted methods for storing facts and thoughts on a variety of different medium. A very efficient way of doing this is using logograms, like Chinese have done for ages. Another method is to represent each syllable in a word by a symbol, like the Japanese do when writing telegrams. However, the most familiar way of storing information is using a limited set of pictures representing so called phonemes. Such a collection is called an alphabet, and often the same glyph is used for different sounds.

Although TEX is primarily meant for typesetting languages that use this third method, in principle the other two can also be dealt with. In this manual we will focus on the languages that use such alphabets.

The little pictures representing the characters that make up an alphabet are more or less standardized, and thereby can be recognized by readers, even if their details differ. Such a collection of pictures, often called glyphs, make up a font.

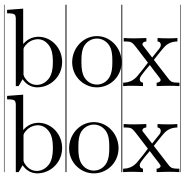


From left to right we see the Computer Modern, a Lucida Bright, a Times Roman and an Antiqua Torunka font, all scaled to 60pt. Fonts colections are designed in such a way that the overall appearance of a page looks good and that reading is as comfortable as possible.



Within a font design there can be variations. In the example above we see a normal, a bold, an italic, and a bold italic alternative of the Lucida Bright font.

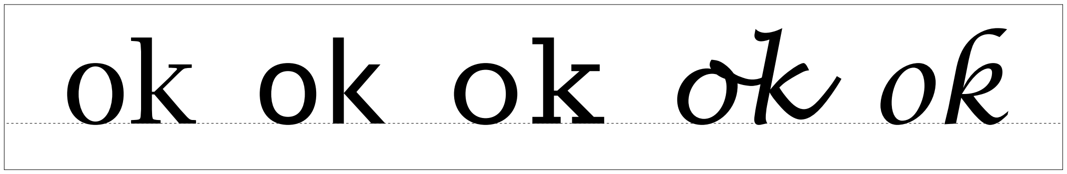
The distance between the individual glyphs in a word depend on the combinations of these glyphs. In the next sample, the gap between the b and the o as well as the distance between the o and the x is slightly altered. This is called kerning.



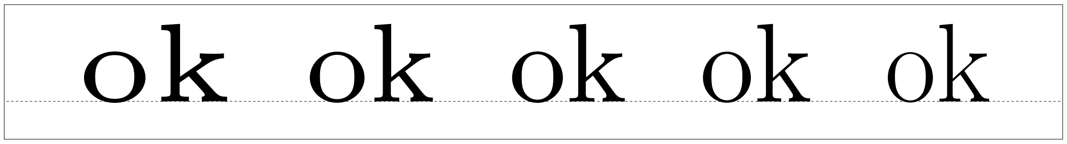
Here we show a Computer Modern, the default TEX font. This font is designed by Donald Knuth and is a variation on a Monotype Times font. The Computer Modern has many kerning pairs, while the Lucida Bright used in this manual has none.

This kind of micro–typography is not to be altered by the user. It is part of the font design. However the user can alter fonts and interline spacing and some more aspects on the level of macro–typography. The choice of font is the main topic of this chapter.

There are different ways to classify fonts. There are classification systems based on times of development, the characteristics of the fonts or the font application, for example in a newspaper or a book.



In this example we see five font styles of Lucida: the Bright, Sans, Typewriter, Handwriting and Calligraphy. This is one of the few examples of a font collection that provides many consistent alternative styles. The Computer Modern is another example of a rather complete font. It is one of the few fonts that comes with dedicated design sizes. The example below shows the differences of a 5, 7, 9, 12 and 17 point design scaled up to 48 points. Such nuances in font size are seldom seen these days.



The general appearance of a style can be classified according to many schemes. In table 5.1 we see some examples of the naming of styles.

Serif	Sans	Mono
Regular	Support	Mono
Roman	Sans	Type

Table 5.1 Some ways of classifying the styles in a font.

The first two series are used by typographers, however in CONTEXT we rather use the last series because it is traditionally used in plain TEX. The command `\rm` is used to switch to a roman/serif/regular style, and `\tt` for switching to mono spaced or typewriter style.

In the next sections we will go into switching of font styles and fonts in your documents. Note that the font switching mechanism is rather complex. This is caused by the different modes like math mode and text mode in CONTEXT. If you want to be able to understand the mechanism you will have to acquaint yourself with the concept of the encoding vector and obtain some knowledge on fonts and their peculiarities.

5.2 The mechanism

Font switching is one of the eldest features of CONTEXT because font switching is indispensable in a macropackage. The last few years extensions to the font switching mechanism were inevitable. We have chosen the following starting points during the development of this mechanism:

- To change a *style* must be easy, this means switching to: roman (serif, regular), sans serif (support), teletype (or monospaced) etc. (`\rm`, `\ss`, `\tt` etc.)
- More than one *variations* of character must be available like slanted and bold (`\sl` and `\bf`).
- Different font *families* like Computer Modern Roman and Lucida Bright must be supported.
- Changing the bodyfont must also be easy, and so font size between 8pt and 12pt must be available by default.
- Within a font different sub- and superscripts must be available. The script sizes can be used during switching of family, style and alternative.

- Specific characteristics of a *body font* like font definition (encoding vector) must be taken into account.

Text can be typeset in different font sizes. We often use the unit `pt` to specify the size. The availability of these font sizes are defined in definition files. Traditionally font designers used to design a glyph collection for each font size, but nowadays most fonts have a design size of 10 points. An exception to this rule is the Computer Modern Roman that comes with most TEX distributions.

The most frequently used font sizes are predefined: 8, 9, 10, 11, 12 and 14.4 points. When you use another size —for example for a titlepage— CONTEXT will define this font itself within the constraints of the used typeface. CONTEXT works with a precision of 1 digit which prevents unnecessary loading of fontsizes with small size differences. When a fontsize is not available CONTEXT prefers to use a somewhat smaller font size. We consider this to be more tolerable than a somewhat bigger font size.

The bodyfont (main font), font style and size is set up with:

```
\setupbodyfont[..,...,..]
...      name serif regular roman sans support sansserif mono type teletype handwritten
         calligraphic 5pt ... 12pt
```

In a running text a temporary font switch is done with the command:

```
\switchtobodyfont[..,...,..]
...      5pt ... 12pt small big global
```

This command doesn't change the bodyfont in headers and footers. With `small` and `big` you switch to a smaller or larger font.

In most cases, the command `\setupbodyfont` is only used once: in the styledefinition. Fontswitching is done with `\switchtobodyfont`. Don't mix these two up because this may lead to some rather strange but legitimate effects.

TEX searches for font information in the file with the extension `tfm`. Pre-loading is possible but CONTEXT will only load these files when necessarry. The reason is that filenames can differ per distribution.

The font used in headers, footers and footnotes are adapted automatically. This includes the interline space and vertical whitespaces. Font switches with `\vi`, `\vii`, `\viii`, `\ix`, `\x`, `\xi` and `\xii` have only local effects.

The commands:

```
{\xii  with these commands \par}
{\xi   for font switching  \par}
{\x    it is possible to   \par}
{\ix   produce an eyetest: \par}
{\viii a x c e u i w m q p \par}
```

When changing the size of the bodyfont, the interline space is adapted automatically. This is shown on the left. On the right we see what happens when the interline space is not adapted.

with these commands
for font switching
it is possible to
produce an eyetest:
a x c e u i w m q p

with these commands
for font switching
it is possible to
produce an eyetest:
a x c e u i w m q p

5.3 Font switching

The mechanism to switch from one style to another is rather complex and therefore hard to explain. To begin with, the terminology is a bit fuzzy. We call a collection of font shapes, like Lucida or Computer Modern Roman a family. Within such a family, the members can be grouped according to characteristics. Such a group is called a style. Examples of styles within a family are: roman, sans serif and `teletype`. We already saw that there can be alternative classifications, but they all refer to the pressence of serifs and the glyphs having equal widths. In some cases handwritten and/or calligraphic styles are also available. Within a style there can be alternatives, like **boldface** and *slanted*.

There are different ways to change into a new a style or alternative. You can use `\ss` to switch to a sans serif font style and `\bf` to get a bold alternative. When a different style is chosen, the alternatives adapt themselves to this style. Often we will typeset the document in one family and style. This is called the bodyfont.

A consequent use of commands like `\bf` and `\sl` in the text will automatically result in the desired bold and slanted altermatives when you change the family or style in the setup area of your input file. A somewhat faster way of style switching is done by `\ssbf`, `\sssl`, etc. but this should be used with care, since far less housekeeping takes place.

The alternatives within a style are given below. The abbreviation `\sl` means *slanted*, `\it` means *italic* and `\bf` means **boldface**. Sometimes `\bs` and `\bi` are also available, meaning ***bold slanted*** and ***bold italic***. When an alternative is not known, ConTEXt will choose a suitable replacement automatically.

With `\os` we tell ConTEXt that we prefer mediaeval or old-style numbers 139 over 139. The `\sc` generates Small Caps. With an `x` we switch to smaller font size, with `a`, `b`, `c` and `d` to a bigger one. The actual font style is stated by `\tf` or typeface.

```
\tfa \tfb \tfc \tfd
\tfx \bfx \slx \itx
\bf \sl \it \bs \bi \sc \os
```

It depends on the completeness of the font definition files whether alternatives like `\bfa`, `\bfb`, etc. are available. Not all fonts have for instance italic and slanted or both their bold alternatives. In such situations, slanted and italic are threated as equivalents.

Switching to a smaller font is accomplished by `\tfx`, `\bfx`, `\slx`, etc., which adapt themselves to the actual alternative. An even more general downscaling is achieved by `\tx`, which adapts itself to the style and alernative. This command is rather handy when one wants to write macros that act like a chameleon. Going one more step smaller, is possible too: `\txx`. Using `\tx` when `\tx` is already given, is equivalent to `\txx`.

Frequent font switching leads to longer processing times. When no sub- or superscripts are used and you are very certain what font you want to use, you can perform fast font switches with: `\rmsl`, `\ssbf`, `\tttf`, etc.

Switching to another font style is done by:

```
\rm \ss \tt \hw \cg
```

When `\rm` is chosen ConTEXt will interpret the command `\tfd` as `\rmd`. All default font setups use `tf`-setups and will adapt automatically.

The various commands will adapt themselves to the actual setup of font and size. For example:

```
{\rm test {\sl test} {\bf test} \tfc test {\tx test} {\bf test}}
{\ss test {\sl test \tx test} {\bf test \tx test}}
```

will result in:

test *test* **test** test test **test**
test *test* *test* **test** **test**

When a character is not available the most acceptable alternative is chosen.

We will not go into the typographical sins of underlining. These commands are discussed in section 11.5 (“Underline”).

5.4 Characters

A number of commands use the parameter `style` to set up the font style and size. You can use commands like `\sl` or `\rma` or keywords like:

```
normal  bold  slanted  boldslanted  italic  bolditalic  type
small  smallbold  smallslanted  ...  smallitalic  ...  smalltype
capital
```

The parameter mechanism is rather flexible so with the parameter `style` you can type `bold` and `\bf` or `bf`. Even the most low level kind of font switching commands like `12ptrmbf` are permitted. This is fast but requires some insight in macros behind this mechanism.

5.5 Available alternatives

There are only a few font families that can handle math. There is the Computer Modern Roman, the very beautiful Lucida Bright that we prefer in electronic documents, and of course one can use the 'prefered by publishers font' Times. These fonts carry a complete set of characters and symbols for mathematical typesetting. Among these, the Computer Modern Roman distinguishes itself by its many design sizes, which pays off when typesetting complicate math. On this design there are a few variations called Euler and Concrete.[16]

The Computer Modern Roman contains 70 charactertypes and sizes. Because a number of charactersizes are not defined the 11 point characters are defined as scaled 9 and 10 point characters under the option `cmr`. With `eul` and `con` we obtain a Computer Modern.

```
\showbodyfont[..,...,..]
...      see p 114: \setupbodyfont
```

With the command `\showbodyfont` an overview is generated of the available characters. Below the 12pt–body font Computer Modern Roman (`cmr`) is shown. The close reader will note that not all alternatives are available by default.

16 See Concrete Mathematics by Knuth cs., an outstanding book from the perspective of typography and didactically.

[cmr,12pt]													
	\tf	\sc	\sl	\it	\bf	\bs	\bi	\tfx	\tfxx	\tfa	\tfb	\tfc	\tfd
\rm	Ag	AG	*Ag*	*Ag*	**Ag**	***Ag***	***Ag***	Ag	Ag	Ag	Ag	Ag	Ag
\ss	Ag	Ag	*Ag*	*Ag*	**Ag**	**Ag**	**Ag**	Ag	Ag	Ag	Ag	Ag	Ag
\tt	Ag	Ag	*Ag*	*Ag*	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag

We can see that the 12pt Lucida Bright (`lbr`) is somewhat bigger than the 12pt Computer Modern Roman. An x–character for example `\bfx` is 2pts smaller than the actual typeface. The bigger characters are scaled by TeX's `\magstep`.

[lbr,12pt]													
	\tf	\sc	\sl	\it	\bf	\bs	\bi	\tfx	\tfxx	\tfa	\tfb	\tfc	\tfd
\rm	Ag	AG	*Ag*	*Ag*	**Ag**	***Ag***	***Ag***	Ag	Ag	Ag	Ag	Ag	Ag
\ss	Ag	Ag	*Ag*	*Ag*	**Ag**	***Ag***	***Ag***	Ag	Ag	Ag	Ag	Ag	Ag
\tt	Ag	Ag	*Ag*	*Ag*	**Ag**	***Ag***	***Ag***	Ag	Ag	Ag	Ag	Ag	Ag

A last remark. When you have chosen a larger charactersize, for example `\tfb`, then `\tf` equals `\tfb`, `\bf` equals `\bfb`, etc. This method is preferable over returning to the original character size.

5.6 Emphasize

Within most macropackages the command `\em` is available. This command behaves like a chameleon which means that it will adapt to the actual typeface. In ConTeXt `\em` has the following characteristics:

- a switch to *slanted* or *italic* is possible
- a switch within `\bf` results in ***bold slanted*** or ***bold italic*** (when available)
- a so called *italic correction* is performed automatically (`\/`)

The bold italic or bold slanted characters are supported only when `\bs` and `\bi` are available.

```
The mnemonic {\em em} means {\em emphasis}.
{\em The mnemonic {\em em} means {\em emphasis}.}
{\bf The mnemonic {\em em} means {\em emphasis}.}
{\em \bf The mnemonic {\em em} {\em emphasis}.}
{\it The mnemonic em {\em means \bf emphasis}.}
{\sl The mnemonic em {\em means \bf emphasis}.}
```

This results in:

The mnemonic *em* means *emphasis*.
The mnemonic em *means* emphasis.
The mnemonic *em* means *emphasis*.
***The mnemonic* em emphasis.**
The mnemonic em means **emphasis**.
The mnemonic em means **emphasis**.

The advantage of the use of `\em` over `\it` and/or `\sl` is that consistent typesetting is enforced.

By default emphasis is set at *slanted*, but in this text it is set at *italic*. The setting is made by:

```
\setupfontenvironment[default][em=italic]
```

5.7 Capitals

Words and abbreviations can be typeset in capitals. Both small and big characters are converted into capitals. When `\cap` is used to typeset a capital the size is that of an `\tx`. When we switch to slanted (`\sl`), bold (`\bf`), etc. the capital letter will also change. Since `\cap` has a specific meaning in math mode, the format implementation is called `\kap`. However in text mode one can use `\cap`.

```
\kap{...}
...     text
```

```
\Cap{...}
...     text
```

```
\CAP{...}
...     text
```

```
\Caps{..  ... ..}
...     text
```

The first command converts all letters to a capital. We advise you not to type capital letters in your source file because real small caps distinguishes between small and big letters.

```
Capitals for \cap {UK} are \cap {OK} and capitals for \cap {USA} are
okay. But what about capitals in \cap {Y2K}.
```

this results in:

Capitals for UK are OK and capitals for USA are okay. But what about capitals in Y2K.

A `\kap` within a `\kap` will not lead to any problems:

```
\kap {People that have gathered their \kap {capital} at the cost of other
people are not seldom \nokap {decapitated} in revolutionary times.}
```

or:

PEOPLE THAT HAVE GATHERED THEIR CAPITAL AT THE COST OF OTHER PEOPLE ARE NOT SELDOM decapitated IN REVOLUTIONARY TIMES.

In this example we see that `\cap` can be temporarily revoked by `\nocap`.

```
\nocap{...}
...     text
```

The command `\Cap` changes the first character of a word into a capital and `\CAP` changes letters that are preceded by `\\` into capital letters. With `\Caps` you can change the first character of several words into a capital letter.

```
\setupcapitals[..,..=..,..]
title    yes no
sc       yes no
```

With this command the capital mechanism can be set up. The key `sc=yes` switches to real SMALL CAPS. With `title` we determine whether capitals in titles are changed.

Next to the former `\cap`-commands we have:

```
\Word{...}
...     text
```

and

```
\Words{..  ... ..}
...     text
```

These commands switch the first characters of words into capitals. All characters in a word are changed with:

```
\WORD{...}
...     text
```

We end this section with real small capitals. When these are available the real small caps `\sc` are preferred over the pseudo-capital in abbreviations and logos.

```
In a manual on \TeX\ and Con\TeX t  there is always the question whether to
type \kap{\TeX} and \kap{Con\TeX t} or {\sc \TeX} and {\sc Con\TeX t}. Both
are defined as a logo in the style definition so we type \type {\TEX} and
\type {\CONTEXT}, which come out as \TEX\ and \CONTEXT.
```

Results in:

In a manual on TEX and ConTEXt there is always the question whether to type TEX and CONTEXT or TEX and CONTEXT. Both are defined as a logo in the style definition so we type `\TEX` and `\CONTEXT`, which come out as TEX and CONTEXT.

IT IS ALWAYS POSSIBLE TO TYPESET TEXT IN SMALL CAPITALS. HOWEVER, REALIZE THAT LOWER CASE CHARACTERS DISCRIMINATE MORE AND MAKE FOR AN EASIER READ.

An important difference between `\kap` and `\sc` is that the last command is used for a specific designed font type. The command `\kap` on the other hand adapts itself to the actual typeface: *KAP*, **KAP**, ***KAP***, etc.

Some typesetting packages stretch words (inter character spacing) to reach an acceptable alignment. In ConTEXt this not supported. On purpose! Words in titles can be stretched by:

```
\stretched{...}
...     text
```

```
\hbox to \hsize {\stretched{there\\is\\much\\stretch\\in ...}}
\hbox to 20em   {\stretched{... and\\here\\somewhat\\less}}
```

With `\\` we enforce a space (`{}` is also allowed).

t h e r e   i s   m u c h   s t r e t c h   i n   . . .
. . . a n d   h e r e   s o m e w h a t   l e s s

These typographically non permitted actions are only allowed in heads. The macros that take care of stretching do this by processing the text character by character.

5.8 Verbatim text

Text can be displayed in verbatim (typed) form. The text is typed between the commands:

```
\starttyping ... \stoptyping
```

Like in:

```
\starttyping
In this text there are enough examples of verbatim text. The command
definitions and examples are typeset with the mentioned commands. Like in
this example.
\stoptyping
```

For in–line typed text the command `\type` is available.

```
\type{...}
...     text
```

A complete file can be added to the text with the command:

```
\typefile{...}
```

...	*file*

The style of typing is set with:

```
\setuptyping[...][..,..=..,..]
```

...	file typing *name*
space	on off
page	yes no
option	slanted normal commands color none
text	yes no
icommand	*command*
vcommand	*command*
ccommand	*command*
before	*command*
after	*command*
margin	*dimension* standard
evenmargin	*dimension*
oddmargin	*dimension*
blank	*dimension* small medium big standard halfline line
escape	*/*
indentnext	yes no
palet	*name* colorpretty
color	*name*

This setup influences the display verbatim (`\starttyping`) and the verbatim typesetting of files (`\typefile`) and buffers (`\typebuffer`). The first optional argument can be used to define a specific verbatim environment.

```
\setuptyping[file][margin=default]
```

When the key `space=on`, the spaces are shown:

```
No␣alignment␣is␣to␣be␣preferred
over␣␣␣aligning␣␣␣by␣␣␣means␣␣of
spaces␣or␣the␣s␣t␣r␣e␣t␣c␣h␣i␣n␣g␣of␣words
```

A very special case is:

```
\definetyping
  [broadtyping]
```

```
\setuptyping
  [broadtyping]
  [oddmargin=-1.5cm,evenmargin=-.75cm]
```

This can be used in:

```
\startbroadtyping
A verbatim line can be very long and when we don’t want to hyphenate we
typeset it in the margin on the uneven pages.
\stopbroadtyping
```

At a left hand side page the verbatim text is set in the margin.

```
A verbatim line can be very long and when we don’t want to hyphenate we
typeset it in the margin on the uneven pages.
```

An in-line verbatim is set up by:

```
\setuptype[..=..]

space     on off
option    slanted normal none
color     name
```

When the parameter `option` is set at `slanted` all text between `<<` and `>>` is typeset in *a slanted letter*. This feature can be used with all parameters. In this way `\type{aa<<bb>>cc}` will result in: aa*bb*cc.

For reasons of readability you can also use other characters than `{` and `}` as *outer* parenthesis. You can choose your own non-active (a non-special) character, for example: `\type+like this+` or `\type-like that-`. Furthermore you can use the mentioned `<<` and `>>`, as in `\type<<like this>>` or even `\type<like that>`.

The parameter `option=commands` enables you to process commands in a typed text. In this option `\` is replaced by `/`. This option is used for typesetting manuals like this one. For example:

```
\seethis <</rm : this command has no effect>>
 /vdots
\sihtees <</sl : neither has this one>>
```

The double `<<` and `>>` overtake the function of `{}`.

Within the type-commands we are using `\tttf`. When we would have used `\tt`, the `\sl` would have produced a slanted and `\bf` a bold typeletter. Now this will not happen:

```
\seethis  : this command has no effect
 ⋮
\sihtees  : neither has this one
```

One of the most interesting options of typesetting verbatim is a program source code. We will limit the information on this topic and refer readers to the documentation in the files `verb-xxx.tex` and `cont-ver.tex`. In that last file you can find the following lines:

```
\definetyping [MP]  [option=MP]
\definetyping [PL]  [option=PL]
\definetyping [JS]  [option=JS]
\definetyping [TEX] [option=TEX]
```

Here we see that it is possible to define your own verbatim environment. For that purpose we use the command:

```
\definetyping[...][..,..=..,..]

...      file typing name
..=..    see p 123: \setuptyping
```

The definitions above couple such an environment to an option.

```
\startMP
beginfig (12) ;
  MyScale = 1.23 ;
  draw unitsquare scaled MyScale shifted (10,20) ;
endfig ;
\stopMP
```

In color (or reduced gray) this will come out as:

```
beginfig (12) ;
  MyScale = 1.23 ;
  draw unitsquare scaled MyScale shifted (10,20) ;
endfig ;
```

These environments take care of typesetting the text in such a way that the typographics match the chosen language. It is possible to write several filters. Languages like METAPOST, METAFONT, PERL, JAVASCRIPT, SQL, and off course TEX are supported. By default color is used to display these sources, where several palettes take care of the different commands. That is why you see the parameter `palet` in `\setuptyping`. One can use font changes or even own commands instead, by assigning the appropriate values to the `icommand` (for identifiers), `vcommand` (for variables) and `ccommand` parameters (for the rest). By default we have:

```
\setuptyping [icommand=\ttsl, vcommand=, ccommand=\tf]
```

We have some alternatives for `\type`. When typesetting text with this command the words are not hyphenated. Hyphenation is performed however when one uses:

```
\typ{...}
...     text
```

When you are thinking of producing a manual on TEX you have two commands that may serve you well:

```
\tex{...}
...     text
```

```
\arg{...}
...     text
```

The first command places a \ in front of typed text and the second command encloses the text with `{}`.

5.9 Math

Many TEX users have chosen TEX for its superb math type setting. The math oriented character of TEX has also influenced the font mechanism. We will not go into any details but the central key is the *family*. There is a font family for `\bf`, `\it`, etc. Within a family we distinguish three members: text, script and scriptscript, or a normal, smaller and smallest font. The normal font size is used for running text and the smaller ones for sub and superscripts. The next example will show what the members of a font family can do.

```
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\rm 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\tf 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\bf 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\sl 6x^2$
```

When this is typeset you see this:

$\mathrm{x}^2 + \mathbf{x}^2 + \mathit{x}^2 + \mathit{x}^2 + \boldsymbol{x}^2 + \boldsymbol{x}^2 = 6\mathrm{x}^2$
$\mathrm{x}^2 + \mathbf{x}^2 + \mathit{x}^2 + \mathit{x}^2 + \boldsymbol{x}^2 + \boldsymbol{x}^2 = 6\mathrm{x}^2$
$\mathrm{x}^2 + \mathbf{x}^2 + \mathit{x}^2 + \mathit{x}^2 + \boldsymbol{x}^2 + \boldsymbol{x}^2 = \mathbf{6x^2}$
$\mathrm{x}^2 + \mathbf{x}^2 + \mathit{x}^2 + \mathit{x}^2 + \boldsymbol{x}^2 + \boldsymbol{x}^2 = \mathit{6x}^2$

We can see that the characters adapt but that the symbols are typeset in the same font. Technically this means that the symbols are set in font family 0 (there are 16 families) and in this case that is default `\tf`.

It can also be done somewhat differently as we will see in the next example. A new command is used: `\mf`, which stands for *math font*. This command takes care of the symbols in such a way that they are set in the actual font.[17]

$\mathrm{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$\mathbf{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$\mathit{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$\boldsymbol{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$\mathit{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$\boldsymbol{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$

You should take into account that TEX typesets a formula as a whole. In some cases this means that setups at the end of the formula have effect at the beginning.

```
$\tf\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bf\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\sl\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bs\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\it\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bi\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
```

The exact location of `\mf` is not that important. We also could have typed:

```
$\bf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = \mf 6x^2$
```

One other aspect of fonts in math mode is the way reserved names like `\sin` and `\cos` are typeset.

```
$\bf x^2 + \hbox{whatever} + \sin(2x)$
```

Unlike plain TEX, the **sin** is also set bold.

[17] We also see a strange visual effect. It seems as if the lines are sloped.

$\mathbf{x}^2 + \mathbf{whatever} + \mathbf{sin}(\mathbf{2x})$

In ConTEXt the 12pt math (Computer Modern) fonts are defined with:

```
\definebodyfont [12pt] [mm]
  [ex=cmex10 at 12pt,
   mi=cmmi12,
   sy=cmsy10 at 12pt]
```

It is possible to use `\tf`, `\bf`, etc. within math mode.

```
\definebodyfont [10pt,11pt,12pt] [mm]
  [tf=Sans          sa 1,
   bf=SansBold      sa 1,
   sl=SansItalic    sa 1,
   ex=MathExtension sa 1,
   mi=MathItalic    sa 1,
   sy=MathSymbol    sa 1]
```

```
\setupbodyfont
```

The example we used before would become:

$\mathsf{x}^2 + \mathbf{whatever} + \mathsf{sin}(\mathsf{2x})$

5.10 Em and Ex

In specifying dimensions we can distinguish physical units like `pt` and `cm` and internal units like `em` and `ex`. These last units are related to the actual fontsize. When you use these internal units in specifying for example horizontal and vertical spacing you don't have to do any recalculating when fonts are switched in the style definition.

Some insight in these units does not hurt. The width of an `em` is not the with of an M, but that of an — (an em–dash). When this glyph is not available in the font another value is used. Table 5.2 shows some examples. We see that the width of a digit is about `.5em`. In Computer Modern Roman a digit is excactly half an em wide.

In most cases we use `em` for specifying width and and `ex` for height. Table 5.3 shows some examples. We see that the height equals the height of a lowercase x.

\tf	\bf	\sl	\tt	\ss	\tfx
|12|	**|12|**	*|12|*	|12|	|12|	|12|
|M|	**|M|**	*|M|*	|M|	|M|	|M|
| |	| |	| |	| |	| |	| |
|—|	|—|	|—|	|---|	|—|	|—|

Table 5.2 The width of an em.

\tf	\bf	\sl	\tt	\ss	\tfx
=x	=**x**	=*x*	=x	=x	=x

Table 5.3 The height of an ex.

5.11 Definitions

This section is meant for curious users or those users that want to do some experimenting on defining fonts. We will not discuss precise definitions of accents and encodings. For these issues we refer to the examples in the source code and the files `font-xxx` *and* `enco-xxx`*.*

Earlier we have seen that within a font family there are different font sizes. The relations between these sizes are defined with:

```
\definebodyfontenvironment
  [12pt]
  [         text=12pt,       Math dimensions: normal dimensions,
          script=9pt,        super- and subscripts and
    scriptscript=7pt,        supersuper- and subsubscripts.
               x=10pt,       Pseudo caps and
              xx=8pt,        nested pseudo caps.
             big=12pt,       In case we switch to big
           small=10pt]       or small.
```

When we use a fontsize that is not predefined in this way ConTeXt applies the same proportions anyhow. You can alter this definition by specifying the parameter `default`. When you want to have a somewhat bigger fontsize you can type:

```
\definebodyfontenvironment [24pt]
```

You can switch to a 12.4 environment, without any specific actions. Within a group these fontdefinitions are temporary. When you use the definitions several times in your document you should type the definitions in the setup area of your source file (style definition) since this can save much runtime.

An overview of the different fontsizes within a family can be summoned with:

```
\showbodyfontenvironment[..,...,..]
...      see p 114: \setupbodyfont
```

For the `lbr` family of fonts this is:

[lbr]						
text	**script**	**scriptscript**	**x**	**xx**	**small**	**big**
20.7pt	14.4pt	12pt	17.3pt	14.4pt	17.3pt	20.7pt
17.3pt	12pt	10pt	14.4pt	12pt	14.4pt	20.7pt
14.4pt	11pt	9pt	12pt	10pt	12pt	14.4pt
12pt	9pt	7pt	10pt	8pt	10pt	14.4pt
11pt	8pt	6pt	9pt	7pt	9pt	12pt
10pt	7pt	5pt	8pt	6pt	8pt	12pt
9pt	7pt	5pt	7pt	5pt	7pt	11pt
8pt	6pt	5pt	6pt	5pt	6pt	10pt
7pt	6pt	5pt	6pt	5pt	5pt	9pt
6pt	5pt	5pt	5pt	5pt	5pt	8pt
5pt	5pt	5pt	5pt	5pt	5pt	7pt
4pt	4pt	4pt	4pt	4pt	4pt	6pt

For all regular fontsizes environments are predefined that fulfill their purpose adequately. However when you want to do some extra defining yourself there is:

```
\setupbodyfontenvironment[...][..,..=..,..]
...      see p 114: \setupbodyfont
..=..    see p 114: \setupbodyfont
```

The real definitions, i.e. the coupling of commands to the font files, can be done in different ways. The most transparant is the font file `font-phv`.

```
\definefontsynonym [Sans]            [Helvetica]
\definefontsynonym [SansBold]        [Helvetica-Bold]
\definefontsynonym [SansItalic]      [Helvetica-Oblique]
\definefontsynonym [SansSlanted]     [Helvetica-Oblique]
\definefontsynonym [SansBoldItalic]  [Helvetica-BoldOblique]
\definefontsynonym [SansBoldSlanted] [Helvetica-BoldOblique]
\definefontsynonym [SansCaps]        [Helvetica]
```

```
\definebodyfont [14.4pt,12pt,11pt,10pt,9pt,8pt,7pt,6pt,5pt] [ss] [default]
```

With `\definefontsynonym` we couple a logical name, like `SansBold` to a font name, like `Helvetica-Bold`. The real coupling is done somewhere else, by default in the file `font-fil`. There you will see:

```
\definefontsynonym [Helvetica-Bold] [hvb] [encoding=texnansi]
```

This is the only location where a system dependent setup is made. When we work under the naming regime of Karl Berry, the next setup would be more obvious (see `font-ber`):

```
\definefontsynonym [Helvetica-Bold] [phvb] [encoding=ec]
```

Coupling fonts in this way has no real limits. It is interesting to look in `font-unk` where different styles are coupled in such a way that they be used interchangeably.

```
\definefontsynonym [Regular] [Serif]
\definefontsynonym [Roman]   [Serif]
```

We see that the basic specification is `Serif`. The default serif fonts are defined with:

```
\definebodyfont [default] [rm]
  [ tf=Serif        sa 1,
   tfa=Serif        sa a,
      ...
    sl=SerifSlanted sa 1,
   sla=SerifSlanted sa a,
      ...]
```

We saw that `\tf` is the default font. Here `\tf` is defined as `Serif sa 1` which means that it is a serif font, scaled to a normal font size. This `Serif` is projected elsewhere on for example `LucidaBright` which in turn is projected on the filename `lbr`.

The kind of all-in-one definitions as shown previously for Helvetica use the `default` settings and enable easy font definitions. This is okay for fonts that come in one design size.

We, like other TEX users, started with the use of Computer Modern Roman fonts. Since these fonts have specific design sizes CONTEXT supports accurate definitions. See the file `font-cmr`:

```
\definebodyfont [12pt] [rm]
  [ tf=cmr12,
   tfa=cmr12 scaled \magstep1,
   tfb=cmr12 scaled \magstep2,
   tfc=cmr12 scaled \magstep3,
   tfd=cmr12 scaled \magstep4,
    bf=cmbx12,
    it=cmti12,
    sl=cmsl12,
    bi=cmbxti10 at 12pt,
    bs=cmbxsl10 at 12pt,
    sc=cmcsc10 at 12pt]
```

We use here the available TEX-specifications `scaled` and `at`, but CONTEXT also supports a combination of both: `sa` (scaled at). For example if we do not want to use the default Helvetica definition we define:

```
\definebodyfont [12pt,11pt,10pt,9pt,8pt] [ss]
  [tf=hv  sa 1.000,
   bf=hvb sa 1.000,
   it=hvo sa 1.000,
   sl=hvo sa 1.000,
  tfa=hv  sa 1.200,
  tfb=hv  sa 1.440,
  tfc=hv  sa 1.728,
  tfd=hv  sa 2.074,
   sc=hv  sa 1.000]
```

The scaling is done in relation to the bodyfont size. In analogy with TEX's `\magstep` we can use `\magfactor`: instead of `sa 1.440` we specify `sa \magfactor2`. Because typing all these numbers is rather tiresome so we replace `1.200` by `a`, etc. The relations between `a` and `1.200` can be set up in the bodyfont environment.

```
\definebodyfont [12pt,11pt,10pt,9pt,8pt] [ss]
  [tf=hv sa 1, tfa=hv sa a, tfb=hv sa b, tfc=hv sa c, tfd=hv sa d]
```

Since font files are used in all interfaces we use English commands. The definitions take place in files with the name `font-xxx.tex`, see for example the file `font-cmr.tex`.

```
\definebodyfont[..,.1.,..][.2.][..,..=..,..]
```

.1.	5pt ... 12pt default
.2.	rm ss tt mm hw cg
tf	*file*
bf	*file*
sl	*file*
it	*file*
bs	*file*
bi	*file*
sc	*file*
ex	*file*
mi	*file*
sy	*file*
ma	*file*
mb	*file*
mc	*file*

The setups `ex`, `mi`, `sy`, `ms`, `mb` and `mc` relate to the math charactersets. The first three we can also find in plain TeX, the last three are necessary in other font families. The symbols and characters in AMS-TeX can also be used in ConTeXt: `\definebodyfont [ams]`. These can be found in `ma` and `mb`.

The `a–d` are not mandatory. As an example we will define a bigger fontsize of `\tf`:

```
\definebodyfont [10pt,11pt,12pt] [rm] [tfe=Regular at 48pt]
\tfe Big Words.
```

This becomes:

Big Words.

This definition brings us to other definitions. It is possible to define a bodyfont in a several ways. We can use classifications like `Regular`, or abstract names like `TimesRoman`, or filenames, like `tir`, or even fancy names like `HeadLetter`.

```
\definebodyfont[HeadLetter][Regular sa 1.2]
```

After these definitions we can use \HeadLetter to switch fonts. It may be necessary to adapt the interline spacing with \setupinterlinespace like this:

```
\HeadLetter \setupinterlinespace text \par
```

For advanced TEX users there is the dimension–register \bodyfontsize. This variable can be used to set fontwidths. The number (rounded) points is available in \bodyfontpoints.

Until now we assumed that an a will become an a during type setting. However, this is not always the case. Take for example ä or æ. This character is not available in every font and certainly not in the Computer Modern Typefaces. Often a combination of characters \"a or a command \ae will be used to produce such a character. In some situation TEX will combine characters automatically, like in fl that is combined to fl and not fl. Another problem occurs in converting small print to capital print and vice versa.

Below you see an example of the texnansi mapping:

```
\startmapping[texnansi]
  \definecasemap 228 228 196  \definecasemap 196 228 196
  \definecasemap 235 235 203  \definecasemap 203 235 203
  \definecasemap 239 239 207  \definecasemap 207 239 207
  \definecasemap 246 246 214  \definecasemap 214 246 214
  \definecasemap 252 252 220  \definecasemap 220 252 220
  \definecasemap 255 255 159  \definecasemap 159 255 159
\stopmapping
```

This means so much as: in case of a capital the character with code 228 becomes character 228 and in case of small print the character becomes character 196.

These definitions can be found in enco-ans. In this file we can also see:

```
\startencoding[texnansi]
  \defineaccent " a 228
  \defineaccent " e 235
  \defineaccent " i 239
  \defineaccent " o 246
  \defineaccent " u 252
  \defineaccent " y 255
\stopencoding
```

and

```
\startencoding[texnansi]
  \definecharacter ae 230
  \definecharacter oe 156
  \definecharacter o  248
  \definecharacter AE 198
\stopencoding
```

As a result of the way accents are placed over characters we have to approach accented characters different from normal characters. There are two methods: TEX does the accenting itself *or* prebuild accentd glyphs are used. The definitions above take care of both methods. Other definitions are sometimes needed. In the documentation of the file `enco-ini` more information on this can be found.

We once again return to font definitions. Fast fontswitching is done with commands like `\xii` or `\twelvepoint`, which is comparable to the way it is done in plain TEX. These commands are defined with:

```
\definefontsynonym [twelvepoint] [12pt]
\definefontsynonym [xii]         [12pt]
```

The keys in `\setupbodyfont` are defined in terms of:

```
\definefontstyle [rm,roman,serif,regular]    [rm]
\definefontstyle [ss,sansserif,sans,support] [ss]
\definefontstyle [tt,teletype,type,mono]     [tt]
\definefontstyle [hw,handwritten]            [hw]
\definefontstyle [cg,calligraphic]           [cg]
```

In many command setups we encounter the parameter `style`. In those situations we can specify a key. These keys are defined with `\definestyle`. The third argument is only of importance in chapter and section titles, where, apart from `\cap`, we want to obey the font used there.

```
\definestyle [normal]                  [\tf]  []
\definestyle [bold]                    [\bf]  []
\definestyle [type]                    [\tt]  []
\definestyle [italic]                  [\it]  []
\definestyle [slanted]                 [\sl]  []
\definestyle [bolditalic,italicbold]   [\bs]  []
\definestyle [boldslanted,slantedbold] [\bs]  []
\definestyle [small,smallnormal]       [\tfx] []
```

In section 5.6 we have already explained how *emphasizing* is defined. With oldstyle digits this is somewhat different. We cannot on the forehand in what font these can be found. By default we have the setup:

```
\definefontsynonym [OldStyle] [MathItalic]
```

As we see they are obtained from the same font as the math italic characters.

In addition to these commands there are others, for example macros for manipulating accents. These commands are discussed in the file `font-ini`. More information can also be found in the file `core-fnt` and specific gimmicks in the file `supp-fun`. So enjoy yourself.

5.12 Page texts

Page texts are texts that are placed in the headers, footers, margins and edges of the so called pagebody. This sentence is for instance typeset in the bodyfont in the running text. The fonts of the page texts are set up by means of different commands. The values of the parameters may be something like `style=bold` but `style=\ss\bf` is also allowed. Setups like `style=\ssbf` are less obvious because commands like `\kap` will not behave the way you expect.

Switching to a new font style (`\ss`) will cost some time. Usually this is no problem but in interactive documents where we may use interactive menus with dozens of items and related font switches the effect can be considerable. In that case a more efficient font switching is:

```
\setuplayout[style=\ss]
```

Border texts are setup by its command and the related key. For example footers may be set up with the key `letter`:

```
\setupfooter[style=bold]
```

5.13 Files

A number of font definition files that are standard in most distributions are mentioned in table 5.4. These fonts can be recalled by their last three letters.

The most commonly used encoding vectors, like `ans`, `ec` and `il2`, are preloaded. Extra encoding files are loaded by `\useencoding`, but this is seldom needed. The last two files mentioned in table 5.5 relate to the support of the non–standard keyboard styles. These should be loaded explicitly.

`font-cmr`	Computer Modern Roman
`font-csr`	Computer Slavik Roman (?)
`font-con`	Concrete Roman
`font-eul`	Euler
`font-ams`	American Mathematics Society
`font-ant`	Antykwa Torunska
`font-lbr`	Lucida Bright
`font-pos`	Base PostScript Fonts
`font-ptm`	Times Roman
`font-phv`	Helvetica
`font-pcr`	Courier
`font-fil`	Standard Filenames
`font-ber`	Karl Berry FileNames

Table 5.4 Some standard font definition files ($pos = ptm + phv + pcr$).

`enco-ans`	TeXnansi
`enco-ec`	European Computer
`enco-il2`	ISO Latin 2
`enco-plr`	Polish Roman
`enco-ibm`	default IBM PC code page
`enco-win`	default MS Windows code page

Table 5.5 Some standard encoding definition files.

5.14 Figures

When you use figures in your document they may contain text. Most of time the TEX–fonts are not available. When you use a serif in your document you can best use a Helvetica in the figures. In figure 5.1 we use a Helvetica, while we use Knuth's Sans Serif in the caption.

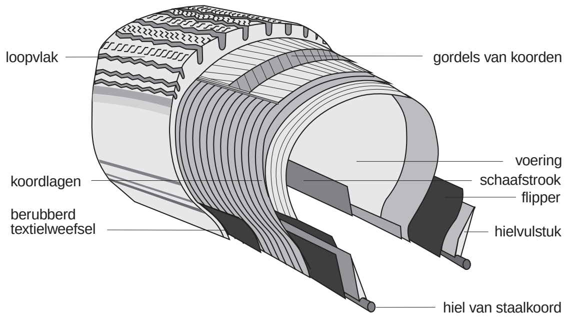


Figure 5.1 The use of fonts in pictures.

Color and background

6

6.1 Introduction

Judicious use of color can enhance your document's layout. For example. in interactive documents color can be used to indicate hyperlinks or other aspects that have no meaning in paper documents, or background colors can be used to indicate screen areas that are used for specific information components.

In this chapter we describe the ConTEXt color support. We will also pay attention to backgrounds and overlays because these are related to the color mechanism.

6.2 Color

One of the problems in typesetting color is that different colors may result in identical gray shades. We did some research in the past on this subject and we will describe the ConTEXt facilities on this matter and the way ConTEXt forces us to use color consistently. Color should not be used indiscriminately, therefore you first have to activate the color mechanism:

```
\setupcolors[state=start]
```

Other color parameters are also available:

```
\setupcolors[..,..=..,..]
```

state	start <u>stop</u> global local
conversion	<u>yes</u> no always
reduction	yes <u>no</u>
rgb	<u>yes</u> no
cmyk	<u>yes</u> no

The parameter `state` can also be set at `local` or `global`. If you do not know whether the use of color will cross a page boundary, then you should use `global` or `start` to keep track of the color. We use `local` in documents where color will never cross a page border, as is the case in many screen documents. This will also result in a higher processing speed. (For most documents it does not hurt that much when one simply uses `start`).

By default both the RGB and CMYK colorspaces are supported. When the parameter `cmyk` is set at `no`, then the CMYK color specifications are automatically converted to RGB. The reverse is done when `rgb=no`. When no color is allowed the colors are automatically converted to weighted grayshades. You can set this conversion with `conversion`. When set to `always`, all colors are converted to gray, when set to `yes`, only gray colors are converted.

Colors must be defined. For some default color spaces, this is done in the file `colo-`*`xxx`*`.tex`. After definition the colors can be recalled with their mnemonic name *xxx*. By default the file `colo-rgb.tex` is loaded. In this file we find definitions like:

```
\definecolor [darkred]   [r=.5, g=.0, b=.0]
\definecolor [darkgreen] [r=.0, g=.5, b=.0]
............ ..........  .................
```

A file with color definitions is loaded with:

```
\setupcolor[rgb]
```

Be aware of the fact that there is also a command `\setupcolors` that has a different meaning. The `rgb` file is loaded by default.

Color must be activated like this:

```
\startcolor[darkgreen]
We can use as many colors as we like. But we do have to take into
account that the reader is possibly \color [darkred] {colorblind}. The
use of color in the running text should always be carefully considered.
The reader easily tires while reading multi||color documents.
\stopcolor
```

In the same way you can define CMYK colors and grayshades:

```
\definecolor [cyan] [c=1,m=0,y=0,k=0]
\definecolor [gray] [s=0.75]
```

gray can also be defined like this:

```
\definecolor [gray] [r=0.75,r=0.75,b=0.75]
```

When the parameter `conversion` is set at `yes` the color definitions are automatically downgraded to the `s`–form: `[s=.75]`. The `s` stands for ‘screen’. When `reduction` is `yes`, the black component of a CMYK color is distilled from the other components.

One of the facillities of color definition is the heritage mechanism:

```
\definecolor [important] [red]
```

These definitions enable you to use colors consistently. Furthermore it is possible to give all important issues a different color, and change colors afterwards or even in the middle of a document.

So, next to `\setupcolors` we have the following commands for defining colors:

```
\definecolor[...][..,..=..,..]
...      name
r        text
g        text
b        text
c        text
m        text
y        text
k        text
s        text
```

A color definition file is loaded with:

```
\setupcolor[...]
...      name
```

Typesetting color is done with:

```
\color[.1.]{.2.}
.1.      text
```

```
\startcolor[...] ... \stopcolor
...      name
...      text
```

A complete palette of colors is generated with:

```
\showcolor[...]
...      name
```

Figure 6.1 shows the colors that are standard available (see `colo-rgb.tex`).

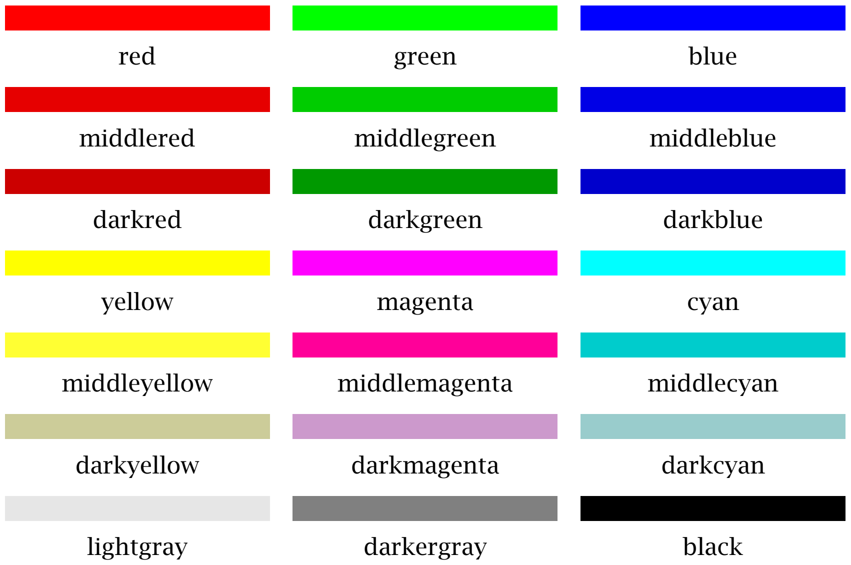


Figure 6.1 Some examples of colors.

The use of color in TeX is not trivial. TeX itself has no color support. Currently color support is implemented using TeX's low level `\mark`'s and `\special`'s. This means that there are some limitations, but in most cases these go unnoticed.

It is possible to cross page boundaries with colors. The headers and footers and the floating figures or tables will stil be set in the correct colors. However, the mechanism is not robust.

In this sentence we use colors within colors. Aesthetically this is bad.

As soon as a color is defined it is also available as a command. So there is a command `\darkred`. These commands do obey grouping. So we can say `{\darkred this is typeset in dark red}`.

There are a number of commands that have the parameter `color`. In general, when a `style` can be set, `color` can also be set.

The default color setup is:

```
\setupcolors [conversion=yes, reduction=no, rgb=yes, cmyk=yes]
```

This means that both colorspaces are supported and that the *k*-component in CMYK colors is maintained. When `reduction=yes`, the *k*-component is ‘reduced’. With `conversion=no` equal color components are converted to gray shades.

6.3 Grayscales

When we print a document on a black and white printer we observe that the differences between somes colors are gone. Figure 6.2 illustrates this effect.

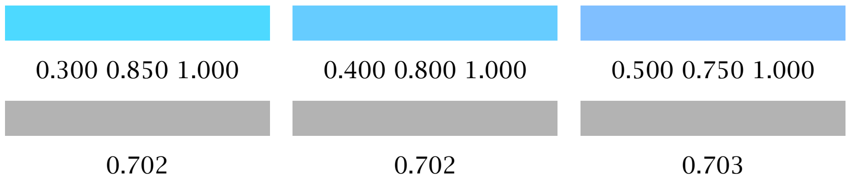


Figure 6.2 Three cyan variations with equal gray shades.

In a black and white print all blocks look the same but the three upper blocks have different cyan based colors. The lower blocks simulate grayshades. We use the following conversion formula:

$$\text{gray} = .30 \times \text{red} + .59 \times \text{green} + .11 \times \text{blue}$$

A color can be displayed in gray with the command:

```
\graycolor[...]
...     text
```

The actual values of a color can be recalled by the commands `\colorvalue{`*name*`}` and `\grayvalue{`*name*`}`.

We can automatically convert all used colors in weighted grayshades.

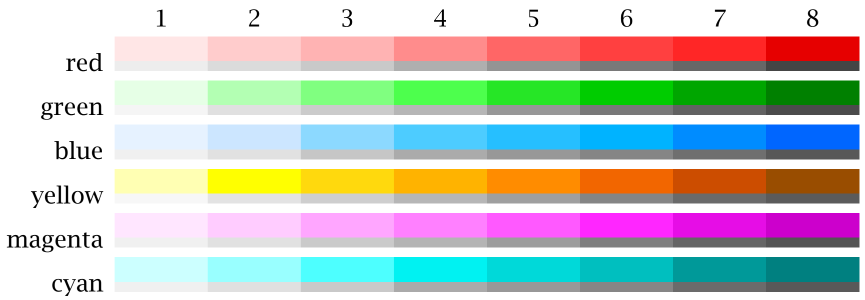
```
\setupcolors [conversion=always]
```

6.4 Colorgroups and palettes

TEX itself has hardly any built-in graphical features. However the CONTEXT color mechanism is designed by looking at the way colors in pictures are used. One of the problems is the effect we described in the last section. On a color printer the picure may look fine, but in black and white the results may be disappointing.

In TEX we can aproach this problem systematically. Therefore we designed a color mechanism that can be compared with that in graphical packages.

We differentiate between individual colors and colorgroups. A colorgroup contains a number of gradations of a color. By default the following colorgroups are defined.



The different gradations within a colorgroup are represented by a number. A colorgroup is defined with:

```
\definecolorgroup[.1.][.2.][..,.3.,..]

.1.     name
.2.     rgb cmyk gray s
.3.     [x:y:z=,..]
```

An example of a part of the RGB definition is:

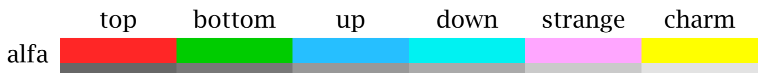
```
\definecolorgroup
  [blue][rgb]
  [1.00:1.00:1.00,
   0.90:0.90:1.00,
   ..............,
   0.40:0.40:1.00,
   0.30:0.30:1.00]
```

The `[rgb]` is not mandatory in this case, because ConTEXt expects RGB anyway. This command can be viewed as a range of color definitions.

```
\definecolor [blue:1] [r=1.00, g=1.00, b=1.00]
\definecolor [blue:2] [r=0.90, g=0.90, b=1.00]
..............
\definecolor [blue:7] [r=0.40, g=0.40, b=1.00]
\definecolor [blue:8] [r=0.30, g=0.30, b=1.00]
```

A color within a colorgroup can be recalled with `name:number`, for example: `blue:4`.

There is no maximum to the number of gradations within a colorgroup, but on the bases of some experiments we advise you to stay within 6 to 8 gradations. We can explain this. Next to colorgroups we have palettes. A pallet consists of a limited number of *logical* colors. Logical means that we indicate a color with a name. An example of a palette is:



The idea behind palettes is that we have to avoid colors that are indistinguishable in black and white print. A palette is defined by:

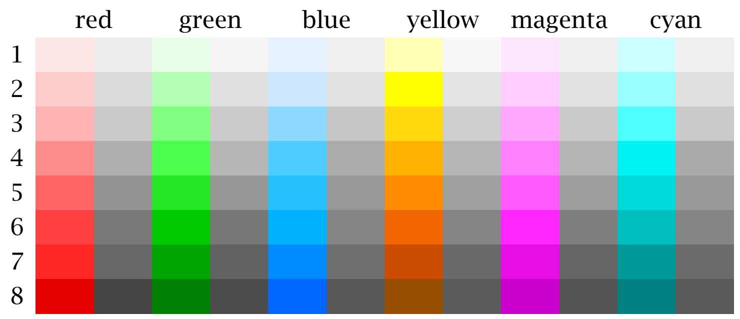
```
\definepalet
  [example]
  [strange=red:3,
       top=green:1,
        .....
    bottom=yellow:8]
```

We define a palette with the command:

```
\definepalet[...][..,..=..,..]

...      name
name     name
```

ConTEXt contains a number of predefined palettes. Within a palette we use the somewhat abstract names of quarks: *top*, *bottom*, *up*, *down*, *strange* and *charm*. There is also *friend* and *rude* because we ran out of names. Be aware of the fact that these are just examples in the RGB definition file and based on our own experiments. Any name is permitted.

The system of colorgroups and palettes is based on the idea that we compose a palette from the elements of a colorgroup with different numbers. Therefore the prerequisite is that equal numbers should have an equal grayshade.



When a palette is composed we can use the command:

```
\setuppalet[...]
...      name
```

After that we can use the colors of the chosen palette. The logical name can be used in for example `\color[strange]{is this not strange}`.

An example of the use of palettes is shown in the verbatim typesetting of TEX code. Within this mechanism colors with names like `prettyone`, `prettytwo`, etc. are used. There are two palettes, one for color and one for gray:

```
\definecolor [colorprettyone] [r=.9, g=.0, b=.0]
\definecolor [grayprettyone]  [s=.3]
```

These palettes are combined into one with:

```
\definepalet
  [colorpretty]
  [  prettyone=colorprettyone,    prettytwo=colorprettytwo,
   prettythree=colorprettythree, prettyfour=colorprettyfour]
\definepalet
  [graypretty]
  [  prettyone=grayprettyone,     prettytwo=grayprettytwo,
   prettythree=grayprettythree,  prettyfour=grayprettyfour]
```

Now we can change all colors by resetting the palette with:

```
\setuptyping[palet=colorpretty]
```

Each filter can be set differently:

```
\definepalet [MPcolorpretty] [colorpretty]
\definepalet [MPgraypretty]  [graypretty]
```

As you can see a palette can inherit its properties from another palette. This example shows something of the color philosophy in ConTEXt: you can treat colors as abstractions and group them into palettes and change these when necessary.

On behalf of the composition of colorgroups and palettes there are some commands available to test whether the colors are distinguishable.

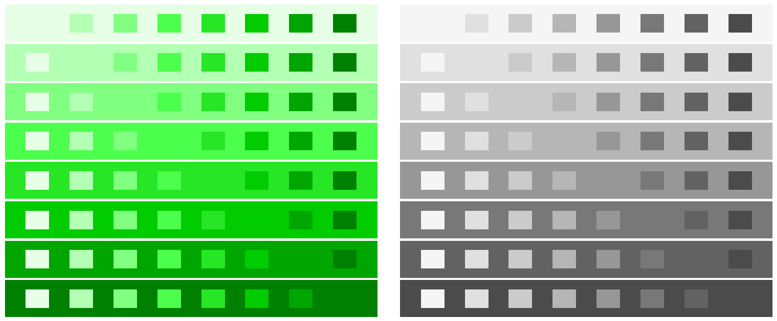
```
\showcolorgroup[.1.][..,.2.,..]

.1.      name
.2.      horizontal vertical name value number
```

```
\showpalet[.1.][..,.2.,..]

.1.      name
.2.      horizontal vertical name value
```

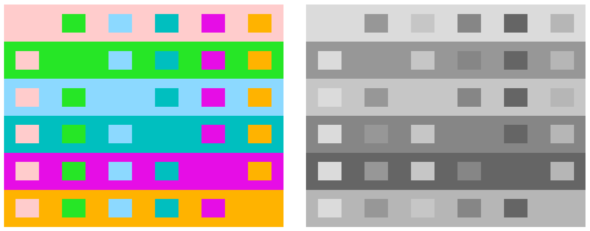
```
\comparecolorgroup[...]

...      name
```

```
\comparepalet[...]

...      name
```

The overviews we have shown thusfar are generated by the first two commands and the gray values are placed below the baseline. On the left there are the colors of the grayshades.



This overview is made with `\comparecolorgroup[green]` and the one below with `\comparepalet[gamma]`.



The standard colorgroups and palettes are composed very carefully and used systematically for coloring pictures. These can be displayed adequately in color and black and white.

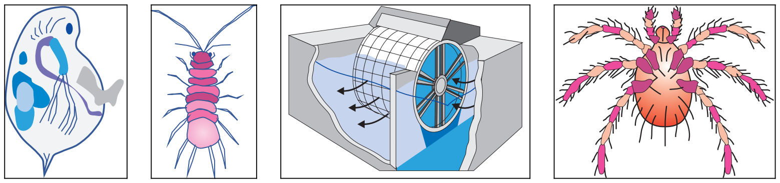


Figure 6.3 Some examples of the use of color.

6.5 Text backgrounds

In a number of commands, for example `\framed`, you can use backgrounds. A background may have a color or a screen (pure gray). By default the `backgroundscreen` is set at `0.95`. Usable values lie between 0.70 and 1.00.

Building screens in TEX is memory consuming and may cause error messages. The screens are therefore build up externally by means of POSTSCRIPT or PDF instructions. This is set up with:

```
\setupscreens[..,..=..,..]

method       dot rule external
resolution   number
factor       number
screen       number
```

The parameter `factor` makes only sense when the method `line` or `dot` is chosen. The parameter `screen` determines the 'grid' of the screen. Text on a screen of 0.95 is still readable.

Visually the TEX screens are comparable with POSTSCRIPT screens. When memory and time are non issues TEX screens come out more beautiful than postscript screens. There are many ways to implement screens but only the mentioned methods are implemented.

Behind the text in the pagebody screens can be typeset. This is done by enclosing the text with the commands:

```
\startbackground
\stopbackground
```

We have done so in this text. Backgrounds can cross page boundaries when necessary. Extra vertical whitespace is added around the text for reasons of readability.

```
\startbackground ... \stopbackground
```

The background can be set up with:

```
\setupbackground[..,..=..,..]

leftoffset     dimension
rightoffset    dimension
topoffset      dimension
bottomoffset   dimension
state          start stop
..=..          see p 264: \setupframed
```

The command `\background` can be used in combination with for example placeblocks:

```
\placetable
  {Just a table.}
  \background
  \starttable[|c|c|c|]
  \HL
  \VL red  \VL green   \VL blue   \VL \AR
  \VL cyan \VL magenta \VL yellow \VL \AR
  \HL
  \stoptable
```

The command `\background` expects an argument. Because a table is 'grouped' it will generate `{ }` by itself and no extra braces are necessary.

```
\background
```

A fundamental difference between colors and screens is that screens are never converted. There is a command `\startraster` that acts like `\startcolor`, but in contrast to the color command, ConTeXt does not keep track of screens across page boundaries. This makes sense, because screens nearly always are used as simple backgrounds.

6.6 Layout backgrounds

In interactive or screen documents the different screen areas may have different functions. Therefore the systematic use of backgrounds may seem obvious. It is possible to indicate all areas or compartments of the pagebody (screenbody). This is done with:

```
\setupbackgrounds[.1.][..,.2.,..][..,..=..,..]

.1.       top header text footer bottom page paper leftpage rightpage
.2.       leftedge leftmargin text rightmargin rightedge
state     start stop repeat
..=..     see p 264: \setupframed
```

Don't confuse this command with `\setupbackground` (singular). A background is only calculated when something has changed. This is more efficient while generating a document. When you want to calculate each background separately you should set the parameter `state` at `repeat`. The page background is always recalculated, since it provides an excellent place for page dependent buttons.

After `\setupbackgrounds` without any arguments the backgrounds are also re-calculated.

A specific part of the layout is identified by means of an axis (see figure 6.4).

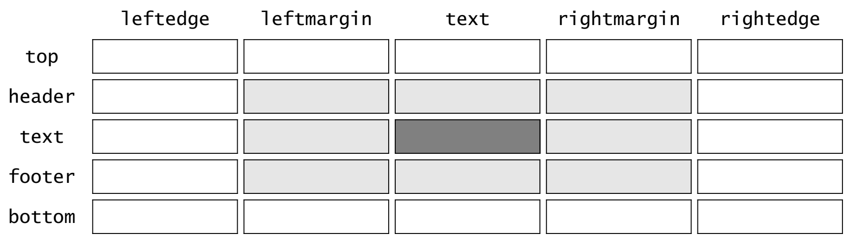


Figure 6.4 The coordinates in `\setupbackgrounds`.

You are allowed to provide more than one coordinate at a time, for example:

```
\setupbackgrounds
  [header,text,footer]
  [text]
  [background=screen]
```

or

```
\setupbackgrounds
  [text]
  [text,rightedge]
  [background=color,backgroundcolor=MyColor]
```

Some values of the paremeter `page`, like `offset` and `corner` also apply to other compartments, for example:

```
\setupbackgrounds
  [page]
  [offset=.5\bodyfontsize
   depth=.5\bodyfontsize]
```

When you use menus in an interactive or screen document alignment is automatically adjusted for offset and/or depth. It is also possible to set the parameter `page` to the standard colors and screens.

If for some reason an adjustment is not generated you can use `\setupbackgrounds` (without an argument). In that case ConTeXt will calculate a new background.

6.7 Overlays

TEX has only limited possibilities to enhance the layout with specific features. In ConTEXt we have the possibility to 'add something to a text element'. You can think of a drawing made in some package or other ornaments. What we technically do is lay one piece of text over another piece text. That is why we speak of 'overlays'.

When we described the backgrounds you saw the paremeters `color` and `screen`. These are both examples of an overlay. You can also define your own background:

```
\defineoverlay[gimmick][\green a green text on a background]
```

```
\framed
  [height=2cm,background=gimmick,align=middle]
  {at\\the\\foreground}
```

This would look like this:

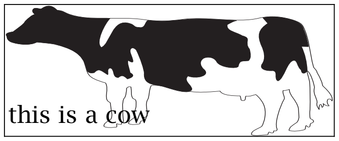
at
the
foreground

a green text on a background

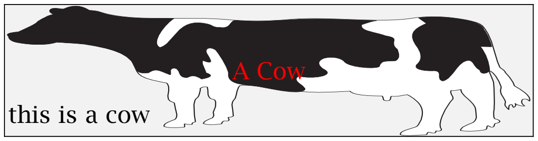
An overlay can be anything:

```
\defineoverlay
  [gimmick]
  [{\externalfigure[koe][width=\overlaywidth,height=\overlayheight]}]
\framed
  [height=2cm,width=5cm,background=gimmick,align=right]
  {\vfill this is a cow}
```

We can see that in designing an overlay the width and height are available in macros. This enables us to scale the figure.



We can combine overlays with one another or with a screen and color.



The TEX definitions look like this:

```
\defineoverlay
  [gimmick]
  [{\externalfigure[koe][width=\overlaywidth,height=\overlayheight]}]
\defineoverlay
  [nextgimmick]
  [\red A Cow]
\framed
  [height=2cm,width=.5\textwidth,
   background={screen,gimmick,nextgimmick},align=right]
  {\vfill this is a cow}
```

6.8 METAPOST

In a CONTEXT document we can use METAPOST code directly. For example:

```
\startMPgraphic
  fill unitsquare scaled 100 withcolor (.2,.3,.4) ;
\stopMPgraphic
```

A direct relation with the CONTEXT color mechanism is obvious:

```
\startMPgraphic
  fill unitsquare scaled 100 withcolor \MPcolor{mark} ;
\stopMPgraphic
```

METAPOST support is very extensive. You can store definitions and re-use them at random. If possible processed METAPOST pictures are re-used.

A detailed discussion on embedding METAPOST graphics is beyond this manual, and therefore will be covered elsewhere. For the moment it is enough to know the basics of putting for instance graphics in the background. In the next example, a graphic is calculated each time it is refered to:

```
\startuseMPgraphic{test a}
  fill unitsquare xscaled \overlaywidth yscaled \overlayheight ;
\stopuseMPgraphic
```

```
\defineoverlay[A Nice Rectangle][\useMPgraphic{test a}]
```

```
\setupbackgrounds[page][background=A Nice Rectangle]
```

When the graphic does not change, we can best reuse it, like:

```
\startreusableMPgraphic{test b}
  fill unitsquare xscaled \overlaywidth yscaled \overlayheight ;
\stopreusableMPgraphic
```

```
\defineoverlay[A Nice Rectangle][\reuseMPgraphic{test b}]
```

```
\setupbackgrounds[page][background=A Nice Rectangle]
```

When using the ConTeXt command line interface TeXexec, graphics are processed automatically. Unless one calls METAPOST at runtime, a second pass is needed to get the graphics in their final state.

Language specific issues

7

7.1 Introduction

One of the more complicated corners of ConTEXt is the department that deals with languages. Fortunately users will seldom notice this, but each language has its own demands and we put quite some effort in making sure that most of the issues on hyphenation rules and accented and non latin characters could be dealt with. For as long as it does not violate the ConTEXt user interface, we also support existing input schemes.

In the early days TEX was very American oriented, but since TEX version 3 there is (simultaneous) support for multiple languages. The input of languages with many accents —sometimes more accents per character— may look rather complicated, depending on the use of dedicated input encodings or special TEX commands.

The situation is further complicated by the fact that specific input does not have a one–to–one relation with the position of a glyph in a font. We discussed this in section 5.11. It is important to make the right choices for input and font encoding.

In this chapter we will deal with hyphenation and language specific labels. More details can be found in the language definition files (`lang-`*xxx*), the font files (`font-`*xxx*) and the encoding files (`enco-`*xxx*). There one can find details on how to define commands that deal with accents and special characters as covered in a previous chapter, sorting indexes, providing support for UNICODE, and more.

7.2 Automatic hyphenating

Each language has its own hyphenation rules. As soon as you switch to another language, ConTEXt will activate the appropriate set of hyphenation patterns for that language. Languages are identified by their official two character identifiers, like: Dutch (`nl`), English (`en`), German (`de`) and French (`fr`). A language is chosen with the following command:[18]

```
\language[...]
...      nl fr en uk de sp cz ..
```

[18] In case of any doubt please check if the hyphenation patterns are included in the `fmt`-file.

Some short cut commands are also available. They can be used enclosed in braces:

```
\nl  \en  \de  \fr  \sp  \uk  \pl  \cz  ...
```

The command `\language[nl]` can be compared with `\nl`. The first command is more transparant. The two character commands may conflict with existing commands. Take, for example, Italian and the code for *italic* type setting. For this reason we use capitals for commands that may cause any conflicts. One may also use the full names, like `czech`.

At any instance you can switch to another language. In the example below we switch from English to French and vice versa.

```
The French composer {\fr Olivier Messiaen} wrote \quote {\fr Quatuor pour
la fin du temps} during the World War II in a concentration camp. This
may well be one of the most moving musical pieces of that period.
```

We use these language switching commands if we cannot be certain that an alternative hyphenation pattern is necessary.

The French compos-
er Olivier Messiaen
wrote ‘Quatuor pour
la fin du temps’ dur-
ing the World War
II in a concentra-
tion camp. This may
well be one of the
most moving musical
pieces of that period.

How far do we go in changing languages. Borrowed words like perestrojka and glasnost are often hyphenated okay, since these are Russian words used in an English context. When words are incorrectly hyphenated you can define an hyphenation pattern with the TEX–command:

```
\hyphenation{ab-bre-via-tion}
```

You can also influence the hyphenation in a text by indicating the allowed hyphenation pattern in the word: at the right locations the command `\-` is added: `al\-lo\-wed`.

7.3 Definitions and setups

When a format file is generated the hyphenation pattern one needs should be added to this file. The definition and installation of a language is therefore not transparant for the user. We show the process to give some insight in the mechanism. An example:[19]

```
\installlanguage
  [en]
  [spacing=broad,
```

[19] The somewhat strange name `\upperleftsinglesixquote` is at least telling us what the quote will look like.

```
   leftsentence=---,
   rightsentence=---,
   leftsubsentence=---,
   rightsubsentence=---,
   leftquote=\upperleftsinglesixquote,
   rightquote=\upperrightsingleninequote,
   leftquotation=\upperleftdoublesixquote,
   rightquotation=\upperrightdoubleninequote,
   date={month,\ ,day,{,\ },year},
   default=en,
   state=stop]
```

and:

```
\installlanguage
  [uk]
  [default=en,
   state=stop]
```

With the first definition you define the language component. You can view this definition in the file `lang-ger.tex`, the german languages. Languages are arranged in language groups. This arrangement is of no further significance at the moment. Since language definitions are preloaded, users should not bother about setting up such files.

The second definition inherits its set up from the English installation. In both definitions `state` is set at `stop`. This means that no patterns are loaded yet. That is done in the files `cont-`*xx*, the language and interface specific ConTeXt versions. As soon as `state` is set at `start`, a new pattern is loaded, which can only be done during the generation of a format file.

We use some conventions in the file names of the patterns `lang-xx.pat` and the exceptions `lang-xx.hyp`. Normally a language is installed with a two character code. However there are three character codes, like deo for hyphenating ‘old deutsch’ and `nlx` the Dutch extended characterset, or 8–bit encoding. On distributions that come with patterns, the filenames mentioned can be mapped onto the ones available on the system. This happens in the file `cont-usr.tex`.

After installation you are not bound to the two character definitions. Default the longer (English) equivalents are defined:

```
\installlanguage[german][de]
```

```
\installlanguage[...][..,..=..,..]
...                 name
spacing             packed broad
lefthyphenmin       dimension
righthyphenmin      dimension
state               start stop
leftsentence        command
rightsentence       command
leftsubsentence     command
rightsubsentence    command
leftquote           command
rightquote          command
leftquotation       command
rightquotation      command
default             name
```

```
\setuplanguage[...][..,..=..,..]
..=..    see p 162: \installlanguage
```

The setup in these commands relate to the situations that are shown below.

```
\currentdate
|<|all right there we go|>|
|<| |<|all right|>| there we go|>|
|<|all right |<|there|>| we go|>|
\quote{all right there we go}
\quotation{all right there we go}
\quotation{\quote{all right} there we go}
\quotation{all right \quote{there} we go}
```

This becomes:

January 27, 2000
—all right there we go—
— —all right— there we go—

—all right —there— we go—
‘all right there we go’
“all right there we go”
“‘all right’ there we go”
“all right ‘there’ we go”

We will discuss `||` in one of the next sections.

7.4 Date

Typesetting a date is also language specific so we have to pay some attention to dates here. When the computer runs at the actual time and date the system date can be recalled with:

```
\currentdate[..,...,..]
...      day month MONTH weekday WEEKDAY year dd mm jj yy d m j y referral
```

The sequence in which `day`, `month` and `year` are given is not mandatory. The pattern `[day,month,year]` results in 27 January 2000. We use `\currentdate[weekday,month,day,{,},year]` to obtain Thursday January 27,2000.

A short cut looks like this: `[dd,mm,yy]` and will result in 270100. Something like `[d,m,y]` would result in 2712000 and with `[referral]` you will get a 20000127. Combinations are also possible. Characters can also be added to the date pattern. The date 27-01-00 is generated by the pattern `[dd,--,mm,--,yy]`.

A date can be (type)set with the command:

```
\date[..,..=..,..][..,...,..]
d        number
m        number
y        number
...      see p 163: \currentdate
```

The first (optional) argument is used to specify the date:

`\date[d=10,m=3,y=1996][weekday,month,day, year]`

When no argument is given you will obtain the actual date. When the second argument is left out the result equals that of `\currentdate`. The example results in:

Sunday March 10 1996

7.5 Labels and heads

In some cases ConTEXt will generate text labels automatically, for example the word **Figure** is generated automatically when a caption is placed under a figure. These kind of words are called textlabels. Labels can be set with the command:

```
\setuplabeltext[...][..=..]
...     nl fr en uk de sp cz ..
name    text
```

Relevant labels are: `table`, `figure`, `chapter`, `appendix` and comparable text elements. An example of such a set up is:

```
\setuplabeltext[en][chapter=Chapter ]
\setuplabeltext[nl][hoofdstuk=Hoofdstuk ]
```

The space after `Chapter` is essential, because otherwise the chapternumber will be placed right after the word Chapter (Chapter1 instead of Chapter 1). A labeltext can recalled with:

```
\labeltext{...}
...     text
```

Some languages, like Chinese, use split labels. These can be passed as a comma separated list, like `chapter={left,right}`.

Titleheads for special sections of a document, like abbreviations and appendices are set up with:

```
\setupheadtext[...][..=..]
...     nl fr en uk de sp cz ..
name    text
```

Examples of titleheads are `Content`, `Tables`, `Figures`, `Abbreviations`, `Index` etc. An example definition looks like:

```
\setupheadtext[content=Content]
```

A header can be recalled with:

```
\headtext{...}
...      text
```

Labels and titleheads are defined in the file `lang-xxx`. You should take a look in these files to understand the use of titleheads and labels.

The actual language that is active during document generation does not have to be the same language that is used for the labels. For this reason next to `\language` we have:

```
\mainlanguage[...]
...      nl fr en uk de sp cz ..
```

When typesetting a document, there is normally one main language, say `\mainlanguage[en]`. A temporary switch to another language is then accomplished by for instance `\language[nl]`, since this does not influence the labels and titles. language.

7.6 Language specific commands

German TEX users are accustomed to entering `"e` and getting ë typeset in return. This and a lot more are defined in `lang-ger` using the compound character mechanism built in ConTEXt. Certain two or three character combinations result in one glyph or proper hyphenation. The example below illustrates this. Some macros are used that will not be explained here. Normally, users can stick to simply using the already defined commands.

```
\startlanguagespecifics[de]
  \installcompoundcharacter "a  {\moveaccent{-.1ex}\"a\midworddiscretionary}
  \installcompoundcharacter "s  {\SS}
  .....
  \installcompoundcharacter "U  {\smashaccent\"U}
  \installcompoundcharacter "Z  {SZ}
  .....
  \installcompoundcharacter "ck {\discretionary {k-}{k}{ck}}
  \installcompoundcharacter "TT {\discretionary{TT-}{T}{TT}}
  .....
  \installcompoundcharacter "‘  {\handlequotation\c!leftquotation}
\stoplanguagespecifics
```

The command `\installcompoundcharacter` takes care of the German type setting, `"a` is converted to ä, `"U` in Ü, `"ck` for the right hyphenation, etc. One can add more definitions, but this will violate portability. In a Polish ConTeXt the `/` is used instead of a `"`.

7.7 Automatic translation

It is possible to translate a text automatically in the actual language. This may be comfortable when typesetting letterheads. The example below illustrates this.

```
\translate[..,..=..,..]
name     text
```

```
It depends on the actual language whether a labeltext is type set in
English {\en as an \translate [en=example, fr=exemple], \fr or in French
as an \translate}.
```

The second command call `\translate` uses the applied values. That is, `\translate` with no options uses the options of the last call to `\translate`.

It depends on the actual language whether a labeltext is type set in English as an example, or in French as an exemple.

7.8 Composed words

Words consisting of two separate words are often separated by an intra word dash, as in x–axis. This dash can be placed between `| |`, for example `|--|`. This command, which does not begin with a \, serves several purposes. When `||` is typed the default intra word dash is used, which is `--`. This dash is set up with:

```
\setuphyphenmark[..=..]
sign     -- --- -  ) ( = /
```

The `| |` is also used in word combinations like (intra)word, which is typed as `(intra|)|word`. The mechanism is not foolproof but it serves most purposes. In case the hyphenation is incorrect you can hyphenate the first word of the composed one by hand: `(in\-tra|)|word`.

input	**normal**	**hyphenated**
`intra||word`	intra–word	in-tra–word
`intra|-|word`	intra-word	in-tra-word
`intra|(|word)`	intra(word)	in-tra(-word)
`(intra|)|word`	(intra)word	(in-tra-)word
`intra|--|word`	intra–word	in-tra–word
`intra|~|word`	intra word	in-tra-word

Table 7.1 Hyphenation of composed words.

The main reason behind this mechanism is that TEX doesn't really know how to hyphenate composed words and how to handle subsentences. TEX know a lot about math, but far less about normal texts. Using this command not only serves consistency, but also makes sure that TEX can break compound words at the right places. It also keeps boundary characters at the right place when a breakpoint is inserted.

7 Composed words

Text elements

8

8.1 Introduction

The core of ConTeXt is formed by the commands that structures the text. The most common structuring elements are chapters and sections. The structure is visualized by means of titles and summarized in the table of contents.

A text can be subdivided in different ways. As an introduction we use the methods of H. van Krimpen, K. Treebus and the Collectief Gaade. First we examine the method of van Krimpen:

1. French title
2. title
3. history & copyright
4. mission
5. preface/introduction
6. . . .
7. list of illustrations
8. acknowledgement
9. errata
10. the content
11. notes
12. literature
13. register(s)
14. colofon

The French title is found at the same spread as the back of the cover, or first empty sheet. In the colofon we find the used font, the names of the typesetter and illustrator, the number of copies, the press, the paper, the binding, etc.

The subdivision of Treebus looks like this:

1. French title
2. titlepage
3. colofon
4. copyright
5. mission
6. preface (1)
7. table of content
8. list of illustrations
9. introduction/preface (2)
10. . . .
11. epilogue
12. appendices
13. summaries
14. notes
15. literature
16. used words
17. addenda
18. register
19. acknowledgement photos
20. (colofon)

In this way of dividing a text the colofon is printed on the back of the titlepage. The first preface is written by others and not by the author.

The last text structure is that of the Collectief Gaade:

1. French title
2. series title
3. title
4. copyright
5. mission
6. blank
7. preface
8. table of content
9. introduction

10. ...
11. appendices
12. notes
13. list of illustrations
14. used words
15. bibliography
16. colofon
17. register

Since there seems to be no standardized way of setting up a document, ConTeXt will only provide general mechanisms. These are designed in such a way that they meet the following specifications:

1. In a text the depth of sectioning seldom exceeds four. However, in a complex manuals more depth can be useful. In paper documents a depth of six may be very confusing for the reader but in electronic documents we need far more structure. This is caused by the fact that a reader cannot make a visual representation of the electronic book. Elements to indicate this structure are necessary to be able to deal with the information.
2. Not every level needs a number but in the background every level is numbered to be able to refer to these unnumbered structuring elements.
3. The names given to the structuring elements must be a logical ones and must relate to their purpose.
4. It is possible to generate tables of contents and registers at every level of the document and they must support complex interactivity.
5. A document will be divided in functional components like introductions and appendices with their respective (typographical) characteristics.
6. The hyphenation of titles must be handled correctly.
7. Headers and footers are supported based on the standard labels used in a document. For example `chapter` in a book and `procedure` in a manual.
8. A ConTeXt user must be able to design titles without worrying about vertical and horizontal spacing, referencing and synchronisation.

These prerequisites have resulted in a heavy duty mechanism that works in the background while running ConTeXt. The commands that are described in the following sections are an example of an implementation. We will also show examples of self designed titles.

8.2 Subdividing the text

A text is divided in chapters, sections, etc. with the commands:

```
\part[ref,..]{...}
...      text
```

```
\chapter[ref,..]{...}
...      text
```

```
\section[ref,..]{...}
...      text
```

```
\subsection[ref,..]{...}
...      text
```

```
\subsubsection[ref,..]{...}
...      text
```

and

```
\title[ref,..]{...}
...      text
```

```
\subject[ref,..]{...}
...      text
```

```
\subsubject[ref,..]{...}
...      text
```

```
\subsubsubject[ref,..]{...}
...      text
```

The first series of commands (`\chapter ...`) generate a numbered head, with the second series the titles are not numbered. There are a few more levels available than those shown above.

level	numbered title	unnumbered title
1	`\part`	
2	`\chapter`	`\title`
3	`\section`	`\subject`
4	`\subsection`	`\subsubject`
5	`\subsubsection`	`\subsubsubject`

Table 8.1 The structuring elements.

By default `\part` generates *no* title because most of the times these require special attention and a specific design. In the background however the partnumbering is active and carries out several initialisations. The other elements are set up to typeset a title.

A structuring element has two arguments. The first argument, the reference, makes it possible to refer to the chapter or section from another location of the document. In chapter 9 this mechanism is described in full. A reference is optional and can be left out.

```
\section{Subdividing a text}
```

ConTeXt generates automatically the numbers of chapters and sections. However there are situations where you want to enforce your own numbering. This is also supported.

```
\setuphead[subsection][ownnumber=yes]
\subsection{399}{The old number}
\subsection[someref]{400}{Another number}
```

In this example an additional argument appears. In the background ConTeXt still uses its own numbering mechanism, so operations that depend upon a consistent numbering still work okay. The extra argument is just used for typesetting the number. This user–provided number does not have to be number, it may be anything, like ABC-123.

399 The old number

400 Another number

You can automatically place titles of chapters, sections or other structuring elements in the header and footer with the marking mechanism. Titles that are too long can be shortened by:

```
\nomarking{...}
...      text
```

For example:

```
\chapter{Influences \nomarking{in the 20th century:} an introduction}
```

The text enclosed by `\nomarking` is replaced by dots in the header or footer. Perhaps an easier strategy is to use the automatic marking limiting mechanism. The next command puts the chapter title left and the section title right in the header. Both titles are limited in length.

```
\setupheadertexts[chapter][section]
\setupheader[leftwidth=.4\hsize,rightwidth=.5\hsize]
```

A comparable problem may occur in the table of contents. In that case we use `\nolist`:

```
\chapter{Influences in the 20th century\nolist{: an introduction}}
```

When you type the command `\\` in a title a new line will be generated at that location. When you type `\crlf` in a title you will enforce a new line only in the table of contents. For example:

```
\chapter{Influences in the 20th century:\crlf an introduction}
```

This will result in a two line title in the table of context, while the title is only one line in the text.

It is possible to define your own structuring elements. Your 'own' element is derived from an existing text element.

```
\definehead[.1.][.2.]
.1.      name
.2.      section
```

An example of a definition is:

```
\definehead[category][subsubject]
```

From this moment on the command `\category` behaves just like `\subsubject`, i.e., `\category` *inherits* the default properties of `\subsubject`. For example, `\category` is not numbered.

A number of characteristics available with `\setuphead` are described in section 8.3. Your own defined structuring elements can also be set up. The category defined above can be set up as follows:

```
\setuphead[category][page=yes]
```

This setup causes each new instance of category to be placed at the top of a new page.

We can also block the sectionnumbering with `\setupheads[sectionnumber=no]`. Sectionnumbering will stop but ConTeXt will continue the numbering on the background. This is necessary to be able to perform local actions like the generating local tables of content.

In defining your own structuring elements there is always the danger that you use existing TeX or ConTeXt commands. It is of good practice to use capitals for your own definitions. For example:

```
\definehead[WorkInstruction][section]
```

8.3 Variations in titles

The numbering and layout of chapters, sections and subsections can be influenced by several commands. These commands are also used in the design of your own heads. We advise you to start the design process in one of the final stages of your document production process. You will find that correct header definitions in the setup area of your source file will lead to a very clean source without any layout commands in the text.

The following commands are at your disposal:

`\setuphead[...][..,..=..,..]`

...	*section*
style	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
numberstyle	normal bold slanted boldslanted type cap small... *command*
number	yes no
ownnumber	yes no
page	left right yes
continue	yes no
header	none empty high nomarking
before	*command*
inbetween	*command*
after	*command*
alternative	normal inmargin middle text
command	\command#1#2
numbercommand	\command#1
textcommand	\command#1
prefix	+ – *text*
placehead	yes no
incrementnumber	yes no *file*
align	left right normal broad
tolerance	verystrict strict tolerant verytolerant stretch
indentnext	yes no
file	*name*
expansion	yes command no

Later we will cover many of the parameters mentioned here. This command can be used to set up one or more heads, while the next can be used to set some common features.

`\setupheads[..,..=..,..]`

sectionnumber	yes number no
alternative	normal margin middle text paragraph
separator	*text*
command	\command#1#2

The number of a title can be set up with:

```
\setupheadnumber[.1.][.2.]
.1.      section
.2.      number +number -number
```

This command accepts absolute and relative numbers, so [12], [+2] and [+]. The relative method is preferred, like:

```
\setuphead[chapter][+1]
```

This command is only used when one writes macros that do tricky things with heads. A number can be recalled by:

```
\headnumber[...]
...      section
```

and/or:

```
\currentheadnumber
```

For example:

```
\currentheadnumber   : 3
\headnumber[chapter] : 8
\headnumber[section] : 8.3
```

When you want to use the titlenumber in calculations you must use the command `\currentheadnumber`. This number is calculated by and available after:

```
\determineheadnumber[...]
...      section
```

When headers and footers use the chapter and section titles they are automatically adapted at a new page. The example below results in going to new right hand side page for each chapter.

```
\setuphead
  [chapter]
  [page=right,
   after={\blank[2*big]}]
```

In extensive documents you can choose to start sections on a new page. The title of the first section however should be placed directly below the chapter title. You can also prefer to start this first section on a new page. In that case you set `continue` at `no`. Figure 8.1 shows the difference between these two alternatives.

```
\setuphead
  [section]
  [page=yes,continue=no,
   after=\blank]
```

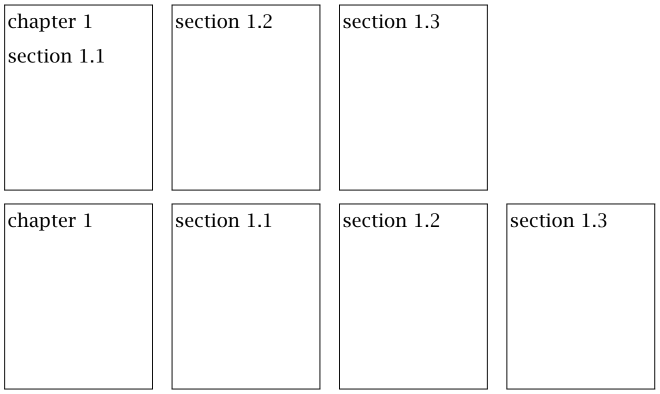


Figure 8.1 Two alternatives for the first section.

It is also possible that you do not want any headers and footers on the page where a new chapter begins. In that case you should set `header` at `empty`, `high`, `nomarking` or an identification of a self defined header (this is explained in section 4.17).

By default the titles are typeset in a somewhat larger font. You can set the text and number style at your own chosen bodyfont. When the titles make use of the same body font (serif, sans, etc.) as the running text you should use neutral identifications for these fonts. So you use `\tfb` instead of `\rmb`. Font switching is also an issue in titles. For example if we use `\ssbf` instead of `\ss\bf` there is a chance that capitals and synonyms are not displayed the way they should. So you should always use the most robust definitions for fontswitching. Commands like `\kap` adapt their behaviour to these switchings.

A chapter title consists of a number and a text. It is possible to define your own command that typesets both components in a different way.

8.3.1 Title alternative equals normal

8.3.2 Title alternative equals inmargin

Title alternative equals middle

These titles were generates by:

```
\setupheads[alternative=normal]
\subsection{Title alternative equals normal}
\setupheads[alternative=inmargin]
\subsection{Title alternative equals inmargin}
\setupheads[alternative=middle]
\subsubject{Title alternative equals middle}
```

In this manual we use a somewhat different title layout. The design of such a title is time consuming, not so much because the macros are complicated, but because cooking up something original takes time. In the examples below we will show the steps in the design process.

```
\def\HeadTitle#1#2%
  {\hbox to \hsize
     {\hfill % the % after {#1} suppresses a space
      \framed[height=1cm,width=2cm,align=left]{#1}%
      \framed[height=1cm,width=4cm,align=right]{#2}}}
```

```
\setuphead[subsection][command=\HeadTitle]
```

8.3.3	**Title**

A reader will expect the title of a section on the left hand side of the page, but we see an alternative here. The title is at the right hand side. One of the advantages of using `\framed` is, that turning `frame=on`, some insight can be gained in what is happening.

8.3.4 Another title

This alternative looks somewhat better. The first definition is slightly altered. This example also shows the features of the command `\framed`.

```
\def\HeadTitle#1#2%
  {\hbox to \hsize \bgroup
   \hfill
   \setupframed[height=1cm,offset=.5em,frame=off]
   \framed[width=2cm,align=left]{#1}%
   \framed[width=4cm,align=right,leftframe=on]{#2}%
   \egroup}

\setuphead
  [subsection]
  [command=\HeadTitle,
   style=\tfb]
```

We see that the font is set with the command `\setuphead`. These font commands should not be placed in the command `\HeadTitle`. You may wonder what happens when ConTeXt encounters a long title. Here is the answer.

8.3.5 | A somewhat longer title

Since we have fixed the height at 1cm, the second line of the title end up below the frame. We will solve that problem in the next alternative. A `\tbox` provides a top aligned box.

```
\def\HeadTitle#1#2%
  {\hbox to \hsize \bgroup
   \hfill
   \setupframed[offset=.5em,frame=off]
   \tbox{\framed[width=3cm,align=left]{#1}}%
   \tbox{\framed[width=4cm,align=right,leftframe=on]{#2}}%
   \egroup}

\setuphead
  [subsection]
  [command=\HeadTitle]
```

This definition results in a title and a number that align on their first lines (due to `\tbox`).

8.3.6 | A consider-ably longer ti-tle

When the title design becomes more complex you have to know more of TeX. Not every design specification can be foreseen.

```
\setuphead[subsubject]   [alternative=text,style=bold]
\setuphead[subsubsubject][alternative=text,style=slantedbold]
```

Titles in the text ***Why are titles in the text more difficult to program in TEX than we may expect beforehand.*** The answer lies in the fact that ConTEXt supports the generation of parallel documents. These are documents that have a printable paper version and an electronic screen version. These versions are coupled and thus hyperlinked by their titles. This means that when you click on a title you will jump to the same title in the other document. So we *couple* document versions:

```
\coupledocument
  [screenversion]
  [repman-e]
  [chapter,section,subsection,subsubsection,part,appendix]
  [The Reporting Manual]
\setuphead
  [chapter,section,subsection,subsubsection,part,appendix]
  [file=screenversion]
```

The first argument in `\coupledocument` identfies the screen document and the second argument specifies the file name of that document. The third argument specifies the coupling and the fourth is a description. After generating the documents you can jump from one version to another by just clicking the titles. This command only preloads references, the actual coupling is achieved by `\setuphead` command. Because titles in a text may take up several lines some heavy duty manipulation is necessary when typesetting such titles as we will see later.

8.4 Meta-structure

You can divide your document in functional components. The characteristics of the titles may depend in what component the title is used. By default we distinguish the next functional components:

- frontmatter
- bodypart
- appendices
- backmatter

Introductions and extroductions are enclosed by `\start ... \stop` constructs. In that case the titles will not be numbered like the chapters, but they are displayed in the table of contents. Within the component 'bodypart' there are no specific actions or layout manipulations, but in the 'appendices' the titles are numbered by letters (A, B, C, etc.).

```
\startfrontmatter
  \completecontent
  \chapter{Introduction}      in content, no number
\stopfrontmatter

\startbodymatter
  \chapter{First}             number 1, in content
    \section{Alfa}            number 1.1, in content
    \section{Beta}            number 1.2, in content
  \chapter{Second}            number 2, in content
    \subject{Blabla}          no number, not in content
\stopbodymatter

\startappendices
  \chapter{Index}             letter A, in content
  \chapter{Abbreviations}     letter B, in content
\stopappendices

\startbackmatter
  \chapter{Acknowlegdement}   no number, in content
  \title{Colofon}             no number, not in content
\stopbackmatter
```

When this code is processed, you will see that commands like `\title` and `\subject` never appear in the table of content and never get a number. Their behaviour is not influenced by the functional component they are used in. The behaviour of the other commands depend on the setup within such a component. Therefore it is possible to adapt the numbering in a functional component with one parameter setup.

8.5 Alternative mechanisms

Not every document can be structured in chapters and sections. There are documents with other numbering mechanisms and other ways to indicate levels in the text. The title mechanism supports these documents.

At the lowest level, the macros of ConTEXt do not work with chapters and sections but with sectionblocks. The chapter and section commands are predefined sectionblocks. In dutch this distinction is more clear, since there we have `\hoofdstuk` and `\paragraaf` as instances of ‘secties’.

```
\definesectionblock[...][..,..=..,..]
...       name
..=..     see p 182: \setupsectionblock
```

```
\setupsectionblock[...][..,..=..,..]
...        name
number     yes no
page       yes right
before     command
after      command
```

```
\definesection[...]
...       name
```

```
\setupsection[...][..,..=..,..]
...               name
conversion        numbers characters Characters romannumerals Romannumerals
previousnumber    yes no
```

By default there are four sectionblocks:

```
\definesectionblock [bodypart]      [headnumber=yes]
\definesectionblock [appendices]    [headnumber=yes]
\definesectionblock [introductions] [headnumber=no]
\definesectionblock [extroductions] [headnumber=no]
```

We see that numbering is set with these commands. When numbering is off local tables of contents can not be generated. When numbers are generated but they do not have to be displayed you can use `\setupheads[sectionnumber=no]`.

By default every sectionblock starts at a new (right hand side) page. This prevents markings from being reset too early. A new page is enforced by `page`.

In ConTEXt there are seven levels in use but more levels can be made available.

```
\definesection [section-1]
\definesection [section-2]
```

```
.............. ..........
\definesection [section-7]
```

There are a number of titles predefined with the command `\definehead`. We show here some of the definitions:

```
\definehead [part]    [section=section-1]
\definehead [chapter] [section=section-2]
\definehead [section] [section=section-3]
```

The definition of a subsection differs somewhat from the others, since the subs inherit the characteristics of a section:

```
\definehead
  [subsection]
  [section=section-4,
   default=section]
```

The definitions of unnumbered titles and subjects are different because we don't want any numbering:

```
\definehead
  [title]
  [coupling=chapter,
   default=chapter,
   incrementnumber=no]
```

The unnumbered title is coupled to the numbered chapter. This means that in most situations the title is handled the same way as a chapter. You can think of the ways new pages are generated at each new unnumbered title or chapter. Characteristics like the style and color are also inherited.

There is more to consider. The predefined sectionblocks are used in appendices, because these have a different numbering system.

```
\setupsection
  [section-2]
  [appendixconversion=Character, % Watch the capital
   previousnumber=no]
\setuphead
  [part]
  [placehead=no]
```

```
\setuphead
  [chapter]
  [appendixlabel=appendix,
   bodypartlabel=chapter]
```

This means that within an `appendix` conversion from number to character takes place, but only at the level of section 2. Furthermore the titles that are related to `section-2` do not get a prefix in front of the number. The prefix consists of the separate numbers of the sectionblocks:

`<section-1><separator><section-2><separator><section-3>` etc.

By default section 2 (appendix) will be prefixed by the partnumber and a separator (.) and this is not desirable at this instance. At that level we block the prefix mechanism and we prevent that in lower levels (section 3 ...) the partnumber is included.

In the standard setup of ConTEXt we do not display the part title. You can undo this by saying:

```
\setuphead[part][placehead=yes]
```

Chapters and appendices can be labeled. This means that the titles are preceded with a word like *Chapter* or *Appendix*. This is done with `\setuplabeltext`, for example:

```
\setuplabeltext[appendix=Appendix~]
```

The look of the titles are defined by `\setuphead`. ConTEXt has set up the lower level section headings to inherit their settings from the higher level. The default setups for ConTEXt are therefore limited to:

```
\setuphead
  [part,chapter]
  [align=normal,
   continue=no,
   page=right,
   head=nomarking,
   style=\tfc,
   before={\blank[2*big]},
   after={\blank[2*big]}]
```

```
\setuphead
  [section]
  [align=normal,
   style=\tfa,
   before={\blank[2*big]},
   after=\blank]
```

With nomarking, we tell ConTEXt to ignore markings in running heads at the page where a chapter starts. We prefer \tfc, because this enables the title to adapt to the actual bodyfont. The { } around \blank are essential for we do not want any conflicts with [ ].

Earlier we saw that new structuring elements could be defined that inherit characteristics of existing elements. Most of the time this is sufficient:

```
\definehead[topic]   [section][style=bold,before=\blank]
\definehead[category][subject][style=bold,before=\blank]
```

One of the reasons that the mechanism is rather complex is the fact that we use the names of the sections as setups in other commands. The marking of category can be compared with that of subject, but that of subject can not be compared with that section. During the last few years it appeared that subject is used for all sorts of titles in the running text. We don't want to see these in headers and footers.

While setting the parameter criterium in lists and registers and the way of numbering, we can choose persection or persubject. For indicating the level we can use the parameter section as well as subject. So we can alter the names of sections in logical ones that relate to their purpose. For example:

```
\definehead [handbook]       [section=section-1]
\definehead [procedure]      [section=section-2]
\definehead [subprocedure]   [section=section-3]
\definehead [instruction]    [procedure]
```

After this we can set up the structuring elements (or inherit them) and generate lists of procedures and instructions. We will discuss this feature in detail in one of the later chapters.

8 Alternative mechanisms

References

9

9.1 Table of contents

The table of contents is very common in books and is used to refer to the text that lies ahead. Tables of content are generated automatically by:

```
\placecontent
```

The table of contents shows a list of chapters and sections but this depends also on the location where the table of contents is summoned. Just in front of a chapter we will obtain a complete table. But just after the chapter we will only obtain a list of relevant sections or subsections. The same mechanism also works with sections and subsections.

```
\chapter{Mammals}
\placecontent
\section{Horses}
```

A table of contents is an example of a combined list. Before discussing combined lists we go into single lists. A single list is defined with:

```
\definelist[.1.][.2.][..,..=..,..]

.1.     name
.2.     name
..=..   see p 189: \setuplist
```

An example of such a definition is:

```
\definelist[firstlevel]
```

Such a list is recalled with:

```
\placelist[firstlevel]
```

Each list may have its own set up:

```
\setuplist[firstlevel][width=2em]
```

Lists can be set up simultaneously, for example:

```
\setuplist[firstlevel,secondlevel][width=2em]
```

To generate a list you type:

```
\placelist[..,...,..][..,..=..,..]
...      name
..=..    see p 189: \setuplist
```

The layout of a list is determined by the values of `alternative` (see table 9.1), `margin`, `width` and `distance`. The alternatives `a`, `b` and `c` are line oriented. A line has the following construct:

margin	width	distance	
	headnumber		head and pagenumber

In a paper document it is sufficient to set up `width`. In an interactive document however the `width` determines the clickable area.[20]

In alternative `d` the titles in the table will be type set as a continuous paragraph. In that case the `before` and `after` have no meaning. The `distance`, that is 1em at a minimum, relates to the distance to the next element in the list. The next set up generates a compact table of contents:

```
\setuplist
  [chapter]
  [before=\blank,after=\blank,style=bold]
\setuplist
  [section]
  [alternative=d,left=(,right=),pagestyle=slanted,prefix=no]
```

Since both lists are defined already when defining the sectioning command, we do not define them here. The parameter `prefix` indicates whether the preceding level indicator numbering is used. In this alternative the prefix is not used. Alternative `d` looks like this:

(1) Table of contents *187* (2) Synonyms *200* (3) Sorting *203* (4) Marking *205* (5) Cross references *208* (6) Predefined references *215* (7) Registers *215*

When `alternative` is set to `d`, an element in the list has the following construction:

left	headnumber	right	head	page	distance

[20] This also depends on the value assigned to `interaction`.

```
\setuplist[..,...,..][..,..=..,..]
```

...	*name*
state	start stop
alternative	a b c ... none command
coupling	on off
criterium	*section* local previous all
pageboundaries	*list*
style	normal bold slanted boldslanted type cap small... *command*
numberstyle	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
pagestyle	normal bold slanted boldslanted type cap small... *command*
color	*name*
command	\command#1#2#3
numbercommand	\command#1
textcommand	\command#1
pagecommand	\command#1
interaction	sectionnumber text pagenumber all
before	*command*
after	*command*
inbetween	*command*
left	*text*
right	*text*
label	yes no
prefix	yes no
pagenumber	yes no
sectionnumber	yes no
aligntitle	yes no
margin	*dimension*
width	*dimension* fit
height	*dimension* fit broad
depth	*dimension* fit broad
distance	*dimension*
separator	*text*
symbol	none 1 2 3 ...
expansion	yes no command
maxwidth	*dimension*
..=..	see p 258: \framed

When you define a title you also define a list. This means that there are standard lists for chapters, sections and subsections, etc. available.

These (sub)sections can be combined into one combined list. The default table of contents is such a combined list:

Table of contents 9

```
\definecombinedlist
  [content]
  [part,
   chapter,section,subsection,subsubsection,
   subsubsubsection,subsubsubsubsection]
  [level=subsubsubsubsection,
   criterium=local]
```

The alternative setups equals that of the separate lists.

```
\definecombinedlist[.1.][..,.2.,..][..,..=..,..]

.1.      name
.2.      list
..=..    see p 190: \setupcombinedlist
```

```
\setupcombinedlist[...][..,..=..,..]

...      name
level    1 2 3 4 section current
..=..    see p 189: \setuplist
```

These commands themselves generate the commands:

```
\completecombinedlist[...][..,..=..,..]

...      name
..=..    see p 190: \setupcombinedlist
```

```
\placecombinedlist[...][..,..=..,..]

...      name
..=..    see p 190: \setupcombinedlist
```

The first command places a title at the top of the list. This title is unnumbered because we do not want the table of contents as an element in the list. In the next section we will discuss lists where the numbered title `\chapter` is used.

alternative	display
a	number – title – pagenumber
b	number – title – spaces – pagenumber
c	number – title – dots – pagenumber
d	number – title – pagenumber (continuous)
e	title (framed)
f	title (left, middle or right aligned)
g	title (centered)

Table 9.1 Alternatives in combined lists.

Possible alternatives are summed up in table 9.1. There are a number of possible variations and we advise you to do some experimenting when you have specific wishes. The three parameters `width`, `margin` and `style` are specified for all levels òr for all five levels separately.

```
\setupcombinedlist
  [content]
  [alternative=c,
   aligntitle=no,
   width=2.5em]
```

The parameter `aligntitle` forces entries with no section number (like titles, subjects and alike) to be typeset onto the left margin. Otherwise the title is aligned to the numbered counterparts (like chapter, section and alike). Compare:

title
12 chapter

with:

title
12 chapter

You can also pass setup parameters to the `\place...` commands. For example:

```
\placecontent[level=part]
```

In this situation only the parts are used in the displayed list. Instead of an identifier, like part or chapter, you can also use a number. However this suggests that you have some insight in the level of the separate sections (part=1, chapter=2 etc.)

A table of contents may cross the page boundaries at an undesired location in the list. Pagebreaking in tables of content can hardly be automated. Therefore it is possible to adjust the pagebreaking manually. The next example illustrates this.

```
\completecontent[pageboundaries={2.2,8.5,12.3.3}]
```

This kind of 'fine-tuning' should be done at the end of the production proces. When the document is revised you have to evaluate the pagebreaking location. ConTeXt produces terminal feedback to remind you when these kind of commands are in effect.

Before a list can be generated the text should be processed twice. When a combined list is not placed after the text is processed twice you probably have asked for a local list.

There are two commands to write something directly to a list. The first command is used to add an element and the second to add a command:

```
\writetolist[.1.]{.2.}{.3.}
.1.     section name
```

```
\writebetweenlist[.1.]{.2.}
.1.     section name
```

We supply a simple example:

```
\writebetweenlist [section] {\blank}
\writetolist      [section] {---} {from here temporary}
\writebetweenlist [section] {\blank}
```

The next command is used in situations where information goes into the title but should not go into the list.

```
\nolist{...}
...     text
```

Consider for example the following example:

```
\definehead[function][ownnumber=yes]
\function{A-45}{manager logistics \nolist{(outdated)}}
\placelist[function][criterium=all]
```

When we call for a list of functions, we will get (...) instead of (outdated). This can be handy for long titles. Keep in mind that each head has a corresponding list.

In an interactive document it is common practice to use more lists than in a paper document. The reason is that the tables of content is also a navigational tool. The user of the interactive document arrives faster at the desired location when many subtables are used, because clicking is the only way to get to that location.

In designing an interactive document you can consider the following setup (probably in a different arrangement):

```
\setuplayout[rightedge=3cm]
\setupinteractions[state=start,menu=on]
\setupinteractionmenu[right][state=start]
\startinteractionmenu[right]
  \placecontent
    [level=current, criterium=previous,
     alternative=f, align=right,
     interaction=all,
     before=, after=]
\stopinteractionmenu
```

These definitions make sure that a table of contents is typeset at every page (screen) in the right edge. The table displays the sections one level deeper than the actual level. So, for each section we get a list of subsections.

When you produce an interactive document with a table of contents at every level you can make a (standard) button that refers to `[previouscontent]`. This reference is generated automatically.

The list elements that are written to a list are not expanded (that is, commands remain commands). When expansion is needed you can set the parameter `expansion`. Expansion is needed in situations where you write variable data to the list. This is seldom the case.

In a more extensive document there may occur situations where at some levels there are no deeper levels available. Then the table of contents at that level is not available either. In that case you need more information on the list so you can act upon it. You can have access to:

`\listlength`	the number of items
`\listwidth`	the maximum width of a list element
`\listheight`	the maximum height of a list element

These values are determined by:

```
\determinelistcharacteristics[..,...,..][..,..=..,..]
...      name
..=..    see p 189: \setuplist
```

We end this section with an overview of the available alternatives. The first three alternatives are primarily meant for paper documents. The `criterium` parameter determines what lists are typeset, so in the next example, the sections belonging to the current chapter are typeset.

```
\placelist
  [section]
  [criterium=chapter,alternative=a]
```

9.1 Table of contents 187
9.2 Synonyms 200
9.3 Sorting 203
9.4 Marking 205
9.5 Cross references 208
9.6 Predefined references 215
9.7 Registers 215

```
\setuplabeltext[en][section={ugh }]
\placelist
  [section]
  [criterium=chapter,alternative=a,
   label=yes,width=2cm]
```

ugh 9.1 Table of contents 187
ugh 9.2 Synonyms 200
ugh 9.3 Sorting 203
ugh 9.4 Marking 205
ugh 9.5 Cross references 208
ugh 9.6 Predefined references 215
ugh 9.7 Registers 215

```
\placelist
  [section]
  [criterium=chapter,alternative=b]
```

9.1 Table of contents 187

9 Table of contents

9.2 Synonyms 200
9.3 Sorting 203
9.4 Marking 205
9.5 Cross references 208
9.6 Predefined references 215
9.7 Registers 215

```
\placelist
  [section]
  [criterium=chapter,alternative=b,
   pagenumber=no,width=fit,distance=1em]
```

9.1 Table of contents
9.2 Synonyms
9.3 Sorting
9.4 Marking
9.5 Cross references
9.6 Predefined references
9.7 Registers

```
\placelist
  [section]
  [criterium=chapter,alternative=c,
   chapternumber=yes,margin=1.5cm]
```

9.1 Table of contents 9-187
9.2 Synonyms 9-200
9.3 Sorting 9-203
9.4 Marking 9-205
9.5 Cross references 9-208
9.6 Predefined references 9-215
9.7 Registers 9-215

```
\placelist % note the spaces on each side of the colon
  [section]
  [criterium=chapter,alternative=c,
   chapternumber=yes,separator={ : },width=fit]
```

9.1 : Table of contents 9-187
9.2 : Synonyms 9-200
9.3 : Sorting 9-203

9.4 : Marking 9–205
9.5 : Cross references 9–208
9.6 : Predefined references 9–215
9.7 : Registers 9–215

```
\placelist
  [section]
  [criterium=chapter,alternative=d]
```

9.1 Table of contents 187 9.2 Synonyms 200 9.3 Sorting 203 9.4 Marking 205
9.5 Cross references 208 9.6 Predefined references 215 9.7 Registers 215

```
\placelist
  [section]
  [criterium=chapter,alternative=d,
   distance=2cm]
```

9.1 Table of contents 187 9.2 Synonyms 200 9.3 Sorting 203
9.4 Marking 205 9.5 Cross references 208 9.6 Predefined
references 215 9.7 Registers 215

```
\placelist
  [section]
  [criterium=chapter,alternative=d,
   left={(},right={)}]
```

(9.1) Table of contents 187 (9.2) Synonyms 200 (9.3) Sorting 203 (9.4) Marking 205
(9.5) Cross references 208 (9.6) Predefined references 215 (9.7) Registers 215

```
\placelist
  [section]
  [criterium=chapter,alternative=e]
```

Table of contents

Synonyms

Sorting

Marking

Cross references

Predefined references

Registers

```
\placelist
  [section]
  [criterium=chapter,alternative=e,
   width=\textwidth,background=screen]
```

Table of contents

Synonyms

Sorting

Marking

Cross references

Predefined references

Registers

```
\placelist
  [section]
  [criterium=chapter,alternative=e,
   width=4cm]
```

Table of contents

Synonyms

Sorting

Marking

Cross references

Predefined references

Registers

```
\placelist
  [section]
  [criterium=chapter,alternative=f]
```

Table of contents

Table of contents 9

Synonyms
Sorting
Marking
Cross references
Predefined references
Registers

```
\placelist
  [section]
  [criterium=chapter,alternative=g]
```

Table of contents
Synonyms
Sorting
Marking
Cross references
Predefined references
Registers

Within a list entry, each element can be made interactive. In most cases, in screen documents, the option all is the most convenient one. Alternative e is rather well suited for screen documents and accepts nearly all parameters of `\framed`. In the next example we use a symbol instead of a sectionnumber. The parameter `depth` applies to this symbol.

```
\placelist
  [section]
  [criterium=chapter,alternative=a,
   pagenumber=no,distance=1em,
   symbol=3,height=1.75ex,depth=.25ex,numbercolor=gray]
```

▨ Table of contents
▨ Synonyms
▨ Sorting
▨ Marking
▨ Cross references
▨ Predefined references
▨ Registers

When using color, don't forget to enable it. In the last example, All alternatives provide the means to hook in commands for the section number, text and pagenumber. Real complete freedom is provided by alternative none.

```
\placelist
  [section]
  [criterium=chapter,alternative=none,
   numbercommand=\framed,
   textcommand=\framed,pagecommand=\framed]
```

9.1 Table of contents 187
9.2 Synonyms 200
9.3 Sorting 203
9.4 Marking 205
9.5 Cross references 208
9.6 Predefined references 215
9.7 Registers 215

```
\def\ListCommand#1#2#3%
  {at page {\bf #3} we discuss {\bf #2}}
```

```
\placelist
  [section]
  [criterium=chapter,alternative=none,
   command=\ListCommand]
```

at page **187** we discuss **Table of contents**
at page **200** we discuss **Synonyms**
at page **203** we discuss **Sorting**
at page **205** we discuss **Marking**
at page **208** we discuss **Cross references**
at page **215** we discuss **Predefined references**
at page **215** we discuss **Registers**

This alternative still provides much of the built–in functionality. Alternative command leaves nearly everything to the macro writer.

```
\def\ListCommand#1#2#3%
  {At p~#3 we discuss {\em #2}; }
```

```
\placelist
  [section]
```

```
[criterium=chapter,alternative=command,
 command=\ListCommand]
```

At p 187 we discuss *Table of contents*; At p 200 we discuss *Synonyms*; At p 203 we discuss *Sorting*; At p 205 we discuss *Marking*; At p 208 we discuss *Cross references*; At p 215 we discuss *Predefined references*; At p 215 we discuss *Registers*;

As an alternative for `none`, we can use `horizontal` and `vertical`. Both commands have their spacing tuned for typesetting lists in for instance menus.

9.2 Synonyms

In many texts we use abbreviations. An abbreviation has a meaning. The abbreviation and its meaning have to be used and typeset consistently throughout the text. We do not like to see ABC and in the next line an ABC. For this reason it is possible to define a list with the used abbreviations and their meanings. This list can be recalled and placed at the beginning or end of a book for the convenience of the reader.

The use of abbreviations is an example of the synonym mechanism. A new category of synonyms is defined with the command:

```
\definesynonyms[.1.][.2.][.3.][.4.]

.1.      singular name
.2.      plural name
.3.      command
.4.      command
```

The way the list is displayed can be influenced by:

```
\setupsynonyms[...][..,..=..,..]

...             name
textstyle       normal bold slanted boldslanted type cap small... command
synonymstyle    normal bold slanted boldslanted type cap small... command
location        left right top serried inmargin inleft inright
width           dimension
state           start stop
criterium       all used
conversion      yes no
expansion       yes command no
command         \command#1#2#3
```

Abbreviations are defined with the command:

```
\definesynonyms[abbreviation][abbreviations][\infull]
```

We will explain the optional fourth argument later. After this definition a new command `\abbreviation` is available. An example of the use of abbreviations is:

```
\abbreviation {UN}  {United Nations}
\abbreviation {UK}  {United Kingdom}
\abbreviation {USA} {United States of America}
```

The meaning can be used in the text by:

```
\infull{abbreviation}
```

It is also possible to add commands in the abbreviation. In that case the command must be typed literally between the `[ ]`:

```
\abbreviation [TEX] {\TeX} {The \TeX\ Typesetting System}
```

Recalling such an abbreviation is done with `\TEX` and the meaning can be fetched with `\infull {TEX}`. In a running text we type `\TEX\` and in front of punctuation `\TEX`.

A synonym is only added to a list when it is used. When you want to display all defined synonyms (used and not used) you have to set the parameter `criterium` at `all`. By setting `state` at `stop` you will prevent list elements to be the added to the list even when they are used. This can be a temporary measure:

```
\setupsynonyms[abbreviation][state=stop]
\abbreviation {NIL} {Not In List}
\setupsynonyms[abbreviation][state=start]
```

Here we left out the optional first argument, in which case the abbreviation itself becomes the command (`\NIL`). So, in this case the next two definitions are equivalent:

```
\abbreviation [NIL] {NIL} {Not In List}
\abbreviation {NIL} {Not In List}
```

The formal definition of a synonym looks like this:

```
\synonym[.1.]{.2.}{.3.}

.1.      text
.2.      text
.3.      text
```

A list of synonyms is generated by:

```
\placelistofsynonyms
```

The next command generates a list with a title (`\chapter`):

```
\completelistofsynonyms
```

Here we see why we typed the plural form during the definition of the synonym. The plural is also used as the title of the list and the first character is capitalized. The title can be altered with `\setuphead` (see section 8.3).

Synonyms are only available after they are used. There are instances when the underlying mechanism cannot preload the definitions. When you run into such troubles, you can try to load the meaning of the synonyms with the command:

```
\loadsynonyms
```

For instance, the meaning of abbreviations can be loaded with `\loadabbreviations`. In order to succeed, the text has to be processed at least once. Don't use this command if things run smoothly.

Next to the predefined abbreviations we also defined the SI–units as synonyms. These must be loaded as a separate module. We will discuss this in section ??.

The attentive reader has seen that the command `\definesynonyms` has four arguments. The fourth argument is reserved for a command with which you can recall the synonym. In this way the synonyms are protected from the rest of the ConTeXt commands and there will be no conflicts using them.

```
\definesynonyms[Function][Functions][\FunctionName][\FunctionNumber]
```

We could define some functions like:

```
\Function [0001] {0001a} {Lithographer}
\Function [0002] {0002x} {Typesetter}
```

Than we can recall number and name by `\FunctionName` (Lithographer and Typesetter) and `\FunctionNumber` (0001a and 0002x), so:

```
The \FunctionName{0001} has functionnumber \FunctionNumber{0001}.
```

9.3 Sorting

Another instance of lists with synonyms is the sorted list. A sorted list is defined with:

```
\definesorting[.1.][.2.][.3.]

.1.      singular name
.2.      plural name
.3.      command
```

The list is set up with:

```
\setupsorting[...][..,..=..,..]

...          name
before       command
after        command
command      \command#1
state        start stop
criterium    all used
style        normal bold slanted boldslanted type cap small... command
expansion    yes command no
```

After the definition the next command is available. The *sort* indicates the name for the list you defined.

```
\sort[.1.]{.2.}

.1.      text
.2.      text
```

In accordance to lists there are two other commands available:

```
\placelistofsorts
```

The title can be set up with `\setuphead`:

```
\completelistofsorts
```

An example of sorting is:

```
\definesorting[city][cities]
\setupsorting[city][criterium=all]

\city {London}
\city {Berlin}
\city {New York}
\city {Paris}
\city {Hasselt}

\placelistofcities
```

The definition is typed in the setup area of your file or in an environment file. The cities can be typed anywhere in your text and the list can be recalled anywhere.

Berlin
Hasselt
London
New York
Paris

Another instance of the sorting command is that where we must type the literal text of the synonym in order to be able to sort the list. For example if you want a sorted list of commands you should use that instance. The predefined command `\logo` is an example of such a list.

```
\logo [TEX]   {\TeX}
\logo [TABLE] {\TaBlE}
```

When you use the alternative with the `[ ]` ConTEXt automatically defines a command that is available throughout your document. In the example above we have `\TABLE` and `\TEX` for recalling the logo. For punctuation we use `\TABLE`.

We advise you to use capital letters to prevent interference with existing ConTEXt and/or TEX commands.

Like in synonyms, a sorted list is only available after an entry is used. When sorting leads to any problems you can load the list yourself:

```
\loadsorts
```

When we add a command in the third argument during the definition of the sorted list we may recall sorted list with this command. In this way the sorted lists can not interfere with existing commands (see section 9.2).

9.4 Marking

There is a feature to add 'invisible' marks to your text that can be used at a later stage. Marks can be used to place chapter or section titles in page headers or footers.

A mark is defined with:

```
\definemarking[.1.][.2.]

.1.      name
.2.      name
```

The second optional argument will be discussed at the end of this section. After the definition texts can be marked by:

```
\marking[.1.]{.2.}

.1.      name
.2.      text
```

and recalled by:

```
\getmarking[.1.][.2.]

.1.      name
.2.      first last previous both all
```

In analogy with the TEX–command `\mark`, we keep record of three other marks per mark (see table 9.2).

marks	**location**
`previous`	the last of the previous page
`first`	the first of the actual page
`last`	the last of the actual page
`both`	first — last
`all`	previous — first — last

Table 9.2 Recorded marks, completed with some combinations.

When you use a combination of marks (`both` and `all`) marks are separated by an —. This separator can be set up with:

```
\setupmarking[...][..=..]

...          name
state        start stop
separator    command
expansion    yes no
```

The use of marks can be blocked with the parameter `state`. The parameter `expansion` relates to the expansion mechanism. By default expansion is inactive. This means that a command is stored as a command. This suits most situations and is memory effective. When you use altering commands in the mark you should activate the expansion mechanism.

Marks are initialised by:

```
\resetmarking[...]

...      name
```

At the beginning of a chapter the marks of sections, subsections, etc. are reset. If we do not reset those marks would be active upto the next section or subsection.

Assume that a word list is defined as follows (we enforce some pagebreaks on purpose):

```
\definemarking[words]

\marking[words]{first}first word ...
\marking[words]{second}second word ...
\page
\marking[words]{third}third word ...
\marking[words]{fourth}fourth word ...
\page
\marking[words]{fifth}fifth word ...
\page
```

The results are shown in table 9.3.

page	previous	first	last
1	—	first	second
2	second	third	fourth
3	fourth	fifth	fifth

Table 9.3 The reordering of marks.

While generating the title of chapters and sections `first` is used. The content of the marks can be checked easily by placing the mark in a footer:

```
\setupfootertexts
  [{\getmarking[words][first]}]
  []
```

or all at once:

```
\setupfootertexts
  [{\getmarking[words][previous]} --
   {\getmarking[words][first]} --
   {\getmarking[words][last]}]
  []
```

A more convenient way of achieving this goal, is the following command. The next method also takes care of empty markings.

```
\setupfootertexts[{\getmarking[words][all]}][]
```

Commands like `\chapter` generate marks automatically. When the title is too long you can use the command `\nomarking` (see section 8.2) or pose limits to the length. In ConTeXt the standard method to place marks in footers is:

```
\setupfootertexts[chapter][sectionnumber]
```

In case you defined your own title with `\definehead`, the new title inherits the mark from the existing title. For example when we define `\category` as follows:

```
\definehead[category][subsection]
```

After this command it does not matter whether we recall the mark by `category` or `subsection`. In this way we can also set up the footer:

```
\setupfootertexts[chapter][category]
```

There are situations where you really want a separate mark mechanism `category`. We could define such a mark with:

```
\definemarking[category]
```

However, we do want to reset marks so we have to have some information on the level at which the mark is active. The complete series of commands would look something like this:

```
\definehead[category][subsection]
\definemarking[category]
\couplemarking[category][subsection]
```

Note that we do this only when we both use category and subsection! After these commands it is possible to say:

```
\setupfootertexts[subsection][category]
```

The command `\couplemarking` is formally defined as:

```
\couplemarking[.1.][.2.]

.1.      name
.2.      name
```

Its counterpart is:

```
\decouplemarking[...]

...      name
```

It is obvious that you can couple marks any way you want, but it does require some insight in the ways ConTEXt works.

9.5 Cross references

We can add reference points to our text for cross referencing. For example we can add reference points at chapter titles, section titles, figures and tables. These reference points are typed between `[ ]`. It is even allowed to type a list of reference points separated by a comma. We refer to these reference points with the commands:

```
\in{.1.}{.2.}[ref]                                        ◀ ▶

.1.      text
```

```
\at{.1.}{.2.}[ref]                                          ◀ ▶
.1.     text
```

```
\about{...}[ref]                                            ◀ ▶
...     text
```

A cross reference to a page, text (number) or both can be made with:

```
\pagereference[ref]
```

```
\textreference[ref]{...}
...     text
```

```
\reference[ref]{...}
...     text
```

The command `\in` provides the number of a chapter, section, figure, table, etc. The command `\at` produces a pagenumber and `\about` produces a complete title. In the first two calls, the second argument is optional, and when given, is put after the number or title.

In the example below we refer to sections and pages that possess reference points:

```
In section~\in[cross references], titled \about[cross references], we
describe how a cross reference can be defined. This section starts
at page~\at[cross references] and is part of chapter~\in[references].
```

This becomes:

In section 9.5, titled “Cross references”, we describe how a cross reference can be defined. This section starts at page 208 and is part of chapter 9.

Here is another variation of the same idea:

```
In \in{section}[cross references], titled \about[cross references], we
describe how a cross reference can be defined. This section starts
at \at{page}[cross references] and is part of \in{chapter}[references].
```

We prefer this way of typing the cross references, especially in interactive documents. The clickable area is in this case not limited to the number, but also includes the preceding word, which is more convenient, especially when the numbering is disabled. In the first example you would have obtained a symbol like◀ that is clickable. This symbol indicates the direction of the cross reference: forward▶ or backward◀.

The direction of a hyperlink can also be summoned by the command `\somewhere`. In this way we find chapters or other text elements before and discuss somewhere later the descriptions.

```
\somewhere{.1.}{.2.}[ref]
.1.     text
```

This command gets two texts. The paragraph will be typed like this:

```
The direction of a hyperlink can also be summoned by the command
\type {\somewhere}. In this way we find chapters or other text elements
\somewhere {before} {after} [text elements] and discuss somewhere
\somewhere {previous} {later} [descriptions] the descriptions.
```

The next command does not need any text but will generate it itself. The generated texts can be defined with `\setuplabeltext` (see page 164).

```
\atpage[ref]
```

At the locations where we make reference points we can also type a complete list of reference points in a comma delimited list:

```
\chapter[first,second,third]{First, second and third}
```

Now you can cross reference to this chapter with `\in[first]`, `\in[second]` or `\in[third]`. In a large document it is difficult to avoid the duplication of labels. Therefore it is advisable to bring some order to your reference point definitions. For example, in this manual we use: `[fig:first]`, `[int:first]`, `[tab:first]` etc. for figures, intermezzos and tables respectively.

CONTEXT can do this for you automatically. Using the command `\setupreferencing`, you can set for instance `prefix=alfa`, in which case all references will be preceded by the word `alfa`. A more memory efficient approach would be to let CONTEXT generate a prefix itself: `prefix=+`. Prefixing can be stopped with `prefix=-`.

In many cases, changing the prefix in many places in the document is not an example of clearness and beauty. For that reason, ConTeXt is able to set the prefix automatically for each section. When for instance you want a new prefix at the start of each new chapter, you can use the command `\setuphead` to set the parameter `prefix` to `+`. The chapter reference itself is not prefixed, so you can refer to them in a natural way. The references within that chapter are automatically prefixed, and thereby local. When a chapter reference if given, this one is used as prefix, otherwise a number is used. Say that we have defined:

```
\setuphead[chapter][prefix=+]
```

```
\chapter[texworld]{The world of \TeX}
```

In this chapter, we can safely use references, without the danger of clashing with references in other chapters. If we have a figure:

```
\placefigure[here][fig:worldmap]{A map of the \TeX\ world}{...}
```

In the chapter itself we can refer to this figure with:

```
\in {figure} [fig:worldmap]
```

but from another chapter, we should use:

```
\in {figure} [texworld:fig:worldmap]
```

In general, when ConTeXt tries to resolve a reference in `\in`, `\at` etc., it first looks to see whether it is a local reference (with prefix). If such a reference is not available, ConTeXt will look for a global reference (without prefix). If you have some trouble understanding the mechanism during document production you can visualize the reference with the command `\version[temporary]`.

There are situations where you want to make a global reference in the middle of document. For example when you want to refer to a table of contents or a register. In that case you can type `-:` in the reference point label that *no* prefix is needed: you type `[-:content]`. Especially in interactive documents the prefix–mechanism is of use, since it enables you to have documents with thousands of references, with little danger for clashes. In the previous example, we would have got a global reference by saying:

```
\placefigure[here][-:fig:worldmap]{A map of the \TeX\ world}{...}
```

The generation of references can be started, stopped and influenced with the command:

```
\setupreferencing[..=..]

state            start stop
sectionnumber    yes no
prefix           + - text
interaction      label text all symbol
width            dimension
left             command
right            command
convertfile      yes no
separator        text
```

In this command the parameter `\sectionnumber` relates to the way the page numbers must be displayed. In interactive documents, we can refer to other documents. In that case, when the parameter `convertfile` is set to `yes`, external filenames are automatically converted to uppercase, which is sometimes needed for CDROM distributions. We will go into details later.

References from another document can be loaded with the command:

```
\usereferences[..,...,..]

...      file
```

With `left` and `right` you can define what is written around a reference generated by `\about`. Default these are quotes. The parameter `interaction` indicates whether you want references to be displayed like *section 1.2*, *section*, *1.2* or as a symbol, like ⧓.

What exactly is a cross reference? Earlier we saw that we can define a reference point by typing a logical label at the titles of chapters, sections, figures, etc. Then we can summon the numbers of chapters, sections, figures, etc. or even complete titles at another location in the document. For some internal purposes the real pagenumber is also available. In the background real pagenumbers play an important role in the reference mechanism.

In the examples below we discuss in detail how the reference point definitions and cross referencing works in CONTEXT.

```
\reference[my reference]{{Look}{at}{this}}
```

The separate elements can be recalled by `\ref`:

`p`	the typeset pagenumber	`\ref[p][my reference]`	212
`t`	the text reference	`\ref[t][my reference]`	Look
`r`	the real pagenumber	`\ref[r][my reference]`	216

s	the subtext reference	`\ref[s][my reference]`	at
e	the extra text reference	`\ref[e][my reference]`	this

In a paper document the reference is static: a number or a text. In an interactive document a reference may carry functionality like hyperlinks. In addition to the commands `\in` and `\at` that we discussed earlier we have the command `\goto`, which allows us to jump. This command does not generate a number or a text because this has no meaning in a paper version.

ConTEXt supports interactivity which is integrated into the reference mechanism. This integration saved us the trouble of programming a complete new set of interactivity commands and the user learns how to cope with these non–paper features in a natural way. In fact there is no fundamental difference in referring to chapter 3, the activation of a JavaScript, referring to another document or the submitting of a completed form.

A direct advantage of this integration is the fact that we are not bound to one reference, but we can define complete lists of references. This next reference is legal:

```
... see \in{section}[flywheel,StartVideo{flywheel 1}] ...
```

As expected this command generates a section number. And in an interactive document you can click on *section nr* and jump to the correct location. At the moment that location is reached a video titled *flywheel 1* is started. In order to reach this kind of comfortable referencing we cannot escape a fully integrated reference mechanism.

Assume that you want to make a cross reference for a general purpose. The name of the reference point is not known yet. In the next example we want to start a video from a general purpose menu:

```
\startinteractionmenu[right]
  \but [previouspage]  previous \\
  \but [nextpage]      next     \\
  \but [ShowAVideo]    video    \\
  \but [CloseDocument] stop     \\
\stopinteractionmenu
```

Now we can activate a video at any given moment by defining `ShowAVideo`:

```
\definerreference[ShowAVideo][StartVideo{a real nice video reel}]
```

This reference can be redefined or erased at any moment:

```
\definereference[ShowAVideo][]
```

```
\definereference[...][ref,..]
...      name
```

```
\startlinenumbering
A special case of referencing is that of referring to linenumbers.
\startline [line:a] Different line numbering mechanism can be used
interchangeably. \startline [line:b] This leads to confusing input.
\stopline [line:a] \startline [line:c] Doesn't it? \stopline [line:c]
\stopline [line:b] A cross reference to a line can result in one line
number or a range of lines. \someline[line:d] {A cross reference is
specified by \type {\inline} where the word {\em line(s)} is
automatically added.} Here we have three cross references: \inline
[line:a], \inline [line:b], \inline[line:c] and \inline {as the last
reference} [line:d].
\stoplinenumbering
```

With `\startlines..\stoplines` you will obtain the range of lines in a cross reference and in case of `\someline` you will get the first line number. In this example we see that we can either let ConTeXt generate a label automatically, or privide our own text between braces.

1 A special case of referencing is that of referring to linenumbers. Different line num-
2 bering mechanism can be used interchangeably. This leads to confusing input. Doesn't
3 it? A cross reference to a line can result in one line number or a range of lines.
4 [redacted][line:d] A cross reference is specified by `\inline` where
5 the word *line(s)* is automatically added. Here we have three cross references: line 1–2, line 2–3,
6 line 2–3 and as the last reference ??.

```
\startlines ... \stoplines
```

```
\someline[ref]
```

```
\inline[ref]
```

9.6 Predefined references

One can imagine that it can be cumbersome and even dangerous for consistency when one has many references which the same label, like **figure** in `\in{figure}[somefig]`. For example, you may want to change each **figure** into `Figure` afterwards. The next command can both save time and force consistency:

```
\definereferenceformat[...][..,..=..,..]

...      name
left     text
right    text
text     text
label    name
```

Given the following definitions:

```
\definereferenceformat [indemo]  [left=(,right=),text=demo]
\definereferenceformat [indemos] [left=(,right=),text=demos]
\definereferenceformat [anddemo] [left=(,right=),text=and]
```

we will have three new commands:

```
\indemo [demo:b]
\indemo {some text} [demo:b]
\indemos {some text} [demo:b] \indemo {and more text} [demo:c]
\indemos [demo:b] \anddemo [demo:c]
```

These will show up as:

demo (BB)
some text (BB)
some text (BB) and more text (CC)
demos (BB) and (CC)

Instead of using the `text` parameter, one can use `label` and recall a predefined label. The parameter `command` can be used to specify the command to use (`\in` by default).

9.7 Registers

A book without a register is not likely to be taken seriously. Therefore we can define and generate one or more registers in CONTEXT. The index entries are written to a separate file. The PERL script TEXUTIL converts this file into a format TEX can typeset.

A register is defined with the command:

```
\defineregister[.1.][.2.]
.1.      singular name
.2.      plural name
```

There are a number of commands to create register entries and to place registers. One register is available by default:

```
\defineregister[index][indices]
```

An entry is created by:

```
\register[.1.]{..+.2.+..}
.1.      text
.2.      text
```

An entry has a maximum of three levels. The subentries are separated by a + or &. We illustrate this with an example.

```
\index{car}
\index{car+wheel}
\index{car+engine}
```

When index entries require special typesetting, for example `\sl` and `\kap` we have to take some measures, because these kind of commands are ignored during list generation and sorting. In those cases we can use the extended version. Between `[ ]` we type the literal ASCII-string which will determine the alphabetical order.

For example we have defined logos or abbreviations like UN, UK and USA (see section 9.2), then an index entry must look like this:

```
\index[UN]{\UN}
\index[UK]{\UK}
\index[USA]{\USA}
```

If we do not do it this way UN, UK and USA will be placed under the \.

A cross reference within a register is created with:

```
\seeregister[.1.]{.2.}{..+.3.+..}
.1.      text
.2.      text
.3.      text
```

This command has an extended version also with which we can input a ‘pure’ literal ASCII string.

A register is generated and placed in your document with:

```
\placeregister[..,..=..,..]                                    ◀ ▶
..=..    see p 218: \setupregister
```

The next command results in register with title:

```
\completeregister[..,..=..,..]                                 ◀ ▶
..=..    see p 218: \setupregister
```

The register can be set up with the command `\setupregister`. When you use the command `\version[temporary]` during processing, the entries and their locations will appear in the margin (see section ??).

```
\setupregister[.1.][.2.][..,..=..,..]
```

.1.	*singular name*
.2.	*name*
n	*number*
balance	yes no
align	yes no
style	normal bold slanted boldslanted type cap small... *command*
pagestyle	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
indicator	yes no
coupling	yes no
sectionnumber	yes no
criterium	*section* local all part
distance	*dimension*
symbol	1 2 ... n a ... none
interaction	pagenumber text
expansion	yes command no
referencing	on off
command	\command#1
location	left middle right
maxwidth	*dimension*
unknownreference	empty none

By default a complete register is generated. However it is possible te generate partial registers. In that case the parameter `criterium` must be set. With `indicator` we indicate that we want a letter in the alphabetical ordering of the entries. When `referencing=on` is a pagereference is generated for every letter indicator, for example `index:a` or `index:w`. We can use these automatically generated references to refer to the page where for instance the a–entries start.

The commands we have mentioned thus far allow us to use a spacious layout in our source file. This means we can type the entries like this:

```
\chapter{Here we are}

\section{Where we are}
\index{here}
\index{where}

Wherever you are ...
```

Between `\chapter` and `\section` we should not type any text because the vertical spacing might be disturbed by the index entries. The empty line after the entry has no consequences.

In case there are problems we always have the option to write index entries to the list by the more direct command:

```
\writetoregister[.1.][.2.]{..+.3.+..}
.1.      singular name
.2.      text
.3.      text
```

There the `expansion` mechanism can be activated. Default expansion is inactive (see page 206).

In this reference manual there is a register with commands. This register is defined and initialised with:

```
\defineregister [macro] [macros]
\setupregister  [macro] [indicator=no]
```

And we can find entries like:

```
\macro{\tex{chapter}}
\macro{\tex{section}}
```

In case we want a register per chapter we can summon the accompanying register with the command below (the command `\tex` will place a \ in front of a word, but is ignored during sorting):[21]

```
\placeregister[macro]
  [criterium=chapter,n=2,before=,after=]
```

and we will obtain:

[21] Of course, `\placemacro` and `\completemacros` are also available.

\getmarking *205*
\in *208*
\inline *208*
\load*sorts* *203*
\load*synonyms* *200*
\logo *203*
\marking *205*
\next*register* *215*
\nolist *187*
\pagereference *208*
\place*combinedlist* *187*
\placelist *187*
\placelistof*sorts* *203*
\placelistof*synonyms* *200*
\place*register* *215*
\ref *208*
\reference *208*
\register *215*
\resetmarking *205*
\see*register* *215*
\setupcombinedlist *187*
\setuplist *187*
\setupmarking *205*
\setupreferencing *208*
\setupregister *215*
\setupsorting *203*
\setupsynonyms *200*
\someline *208*
\somwhere *208*
\sorteer *203*
\startline *208*
\startregister *215*
\synonym *200*
\textreference *208*
\writebetweenlist *187*
\writetolist *187*
\writeto*register* *215*

A warning is due. The quality of the content of a register is completely in your hands. A bad selection of index entries leads to an inadequate register that is of no use to the reader.

Every entry shows one or more pagenumbers. With `symbol` we can define some alternatives. With `distance` the horizontal spacing between word and number or symbol is set.

symbol	**display**
`a`	a b c d
`n`	1 2 3 4
`1`	• • • •
`2`	▬ ▬ ▬ ▬

Table 9.4 Alternatives for pagenumbers in registers.

Most of the time the layout of a register is rather simple. Some manuals may need some form of differentiating between entries. The definition of several registers may be a solution. However the layout can contribute to a better use of the register:

```
\index          {entry}
\index[key]     {entry}
\index[form::]  {entry}
\index[form::key]{entry}
\index          {form::entry}
\index[key]     {form::entry}
\index[form::]  {form::entry}
\index[form::key]{form::entry}
```

The first two alternatives are known, but the rest is new and offers some control over the way the entry itself is typeset. The specification between `[ ]` relates to the pagenumber, the specification in front of the entry relates to the entry itself.

```
\setupregister[index][form][pagestyle=bold,textstyle=slanted]
```

Without any problems we can use different appearances for pagenumber and entry.

```
\setupregister[index][nb][pagestyle=bold]
\setupregister[index][hm][pagestyle=slanted]
```

With for example:

```
\index[nb::]{squareroot}
\index[hm::root]{$\srqt{2}$}
```

The index entries we have discussed so far indicate the one page where the entry is made, but we can also indicate complete ranges of pages using:

```
\startregister[.1.]{..+.2.+..}

.1.      text
.2.      text
```

The entries in between, which are of the same order, are not placed in the register.

```
\startregister[endless]{endless}
...... an endless story ......
\stopregister[endless]
```

An extensive index entry, i.e. an entry with a large number of appearances, may have an uncomfortably long list of pagenumbers. Especially in interactive documents this leads to endless back and forth clicking. For this purpose we designed the feature of linked index entries. This means that you can couple identical entries into a list that enables the user to jump from entry to (identical) entry without returning to the register. The coupling mechanism is activated by:

```
\setupregister[index][coupling=yes]
```

In this way a mechanism is activated that places references in the register (◀⧓▶) as well as in the text (◀ **word** ▶) depending on the availability of alternatives. A jump from the register will bring you to the first, the middle or the last appearance of the entry.

This mechanism is only working at the first level; subentries are ignored. Clicking on the word itself will bring you back to the register. Because we need the clickable word in the text we use the following command for the index entry itself:

```
\coupledregister[.1.]{.2.}

.1.     text
.2.     text
```

For example `\coupledindex{where}`. The couplings must be loaded with the command:

```
\coupleregister[...]

...     name
```

Normally this command is executed automatically when needed, so it’s only needed in emergencies.

Descriptions

10

10.1 Introduction

In a document we can find text elements that bring structure to a document. We have already seen the numbered chapter and section titles, but there are more elements with a recognizable layout. We can think of numbered and non–numbered definitions, itemizations and citations. One of the advantages of TEX and therefore of CONTEXT is that coding these elements enables us to guarantee a consistent design in our document, which in turn allows us to concentrate on the content of our writing.

In this chapter we will discuss some of the elements that will bring structure to your text. We advise you to experiment with the commands and their setups. When applied correctly you will notice that layout commands in your text are seldom necessary.

10.2 Definitions

Definitions of concepts and/or ideas, that are to be typeset in a distinctive way, can be defined by `\definedescription`.

```
\definedescription[...][..,..=..,..]

...      name
..=..    see p 227: \setupdescriptions
```

The first argument of this command contains the name. After the definition a new command is available.

```
\description{.1.}.2.\par

.1.      text
.2.      text
```

An example of the definition is:

```
\definedescription[definition][location=top,headstyle=bold]

\definition{icon}
```

```
An icon is a representation of an action or the name of a computer
program. Icons are frequently used in operating systems on several
computer platforms. \par
```

Several alternatives are displayed below:

icon

An icon is a representation of an action or the name of a computer program. Icons are frequently used in operating systems on several computer platforms.

Some users of those computer platforms are using these icons with an almost religious fanaticism. This brings the word icon almost back to its original meaning. **icon**

icon An icon should be recognizable for every user but they are designed within a cultural and historical setting. In this fast and ever changing era the recognizability of icons is relative.

icon The 8–bit principle of computers was the reason that non–Latin scriptures were hardly supported by the operating systems. Not long ago this changed.

icon What for some languages looked like a handicap has now become a feature. Thousands of words and concepts are already layed down in characters. These characters therefore can be considered icons.

icon It is to be expected that people with expressive languages overtake us in computer usage because they are used to thinking in concepts.

icon The not–so–young generation remembers the trashcan in the earlier operating systems used to delete files. We in Holland were lucky that the text beneath it said: trashcan. A specific character for the trashcan would have been less sensitive misinterpretation, than the rather American–looking garbage receptacle unknown to many young people.

In the fifth example the definition is placed `serried` and defined as:

```
\definedescription
  [definition]
  [location=serried,headstyle=bold,width=broad,sample={icon}]
```

```
\definition{icon}
```

```
What for some languages looked like a handicap has now become a feature.
Thousands of words and concepts are already layed down in characters.
These characters therefore can be considered icons. \par
```

In the seventh example we have set `hang` at `broad`. This parameter makes only sense when we set the label at the right or left. When we set `width` at `fit` or `broad` instead of a number, the width of the sample is used. With `fit`, no space is added, with `broad`, a space of `distance` is inserted. When no sample is given the with of the defined word is used. The parameter `align` specifies in what way the text is aligned. When the definition is placed in the margin or typeset in a serried format, the parameter `margin` is of importance. When set to `standard` or `ja`, the marging follows the document setting. Alternatively you can pass a dimension.

Some characteristics of the description can be specified with:

```
\setupdescriptions[..,...,..][..,..=..,..]
```

...	*name*
headstyle	normal bold slanted boldslanted type cap small... *command*
style	normal bold slanted boldslanted type cap small... *command*
color	*name*
width	fit broad *dimension*
distance	*dimension*
sample	*text*
text	*text*
align	left middle right
margin	standard yes no *dimension*
location	left right top serried inmargin inleft inright hanging
hang	fit broad *number*
before	*command*
inbetween	*command*
after	*command*
indentnext	yes no

The setup of a description can be changed with the command below. This has the same construct as `\definedescription`:

`\setupdescriptions[`*name*`][`*setups*`]`

When a description consists of more than one paragraph, use:

```
\startdescription{...} ... \stopdescription
...      text
```

```
\startdefinition{icon}
An icon is a painting of Jesus Christ, Mother Mary or other holy figures.
These paintings may have a special meaning for some religious people.
```

```
For one reason or the other the description icon found its way to the
computer world where it leads its own life.
```

```
\stopdefinition
```

These commands will handle empty lines adequately.

10.3 Enumeration

Sometimes you will encounter text elements you would like to number, but they do not fit into the category of figures, tables, etc. Therefore ConTeXt has a numbering mechanism that we use for numbering text elements like questions, remarks, examples, etc. Such a text element is defined with:

```
\defineenumeration[..,.1.,..][.2.][..,..=..,..]

.1.      name
.2.      name
..=..    see p 230: \setupenumerations
```

After such a definition, the following commands are available:

```
\name
\subname
\subsubname
\subsubsubname
```

Where `name` stands for any chosen name.

```
\enumeration...\par                                        ◀ ▶

...      text
```

The numbering can take place at four levels. Conversion is related to the last level. If you specify a text, then this will be a label that preceeds every generated number. A number can be set and reset with the command:

```
\setenumeration{value}
\resetenumeration
```

You can use the `start` parameter in the setup command to explictly state a startnumber. Keep in mind that the enumeration commands increase the number, so to start at 4, one must set the number at 3. Numbers and subnumbers and be explictly increased with the commands:

```
\nextenumeration
\nextsubenumeration
\nextsubsubenumeration
```

The example below illustrates the use of `\enumeration`. After the shown commands the content of a remark can be typed after `\remark`.

```
\defineenumeration
  [remark]
  [location=top,
   text=Remark,
   between=\blank,
   before=\blank,
   after=\blank]
```

Some examples of remarks are:

Remark 1

After definition the 'remark' is available at four levels: `\remark`, `\subremark`, `\subsubremark` and `\subsubsubremark`.

Remark 2

This command looks much like the command `\definedescription`.

The characteristics of numbering are specified with `\setupenumerations`. Many parameters are like that of the descriptions because numbering is a special case of descriptions.

```
\setupenumerations[name][setups]
```

`\setupenumerations[..,...,..][..,..=..,..]`

...	*name*
..=..	see p 225: \definedescription
location	left right top serried inmargin inleft inright hanging
text	*text*
levels	*number*
conversion	numbers characters Characters romannumerals Romannumerals
way	bytext by*section*
blockway	yes no
sectionnumber	yes number no
separator	*text*
stopper	*text*
coupling	*name*
couplingway	global local
number	no *name*
aligntitle	no yes
start	*number*

The characteristics of sub and subsub enumerations can be set too. For example:

```
\setupenumerations[example][headstyle=bold]
\setupenumerations[subexample][headstyle=slanted]
```

Just like the description command there is a `\start`–`\stop` construction for multi paragraph typesetting.

`\start`*`enumeration`*` ... \stop`*`enumeration`*

Sometimes the number is obsolete. For example when we number per chapter and we have only *one* example in a specific chapter. In that case you can indicate with a `[-]` that you want no number to be displayed.

Remark

Because this remark was recalled by `\remark[-]` there is *no* number. Just as with other commands, we can also pass a reference label between `[ ]`. Also, we can setup the enumeration to stop numbering by setting `number` to `no`.

The numbering command can be combined usefully with the feature to move textblocks. An example is given in section 12.4. In that example we also demonstrate how to couple one numbered text to another. These couplings only have a meaning in interactive documents where cross references (hyperlinks) can be useful.

The numbering of text elements can appear in different forms. In that case we can let one numbered text element inherit its characteristic from another. We illustrate this in an example.

```
\defineenumeration[first]

\first  The numbering \type {first} is unique. We see that one
argument is sufficient. By default label and number are placed at the left
hand side.

\defineenumeration[second][first][location=right]

\second  The \type {second} inherits its counters from \type {first},
but is placed at the right hand side. In case of three arguments the first
one is the copy and the second the original.

\doornummeren[third,fourth][location=inright]

\third The numbered elements \type {third} and \type {fourth} are both
unique and are placed in right margin.

\fourth  Both are defined in one command but they do have own
counters that are in no way coupled.

\doornummeren[fifth,sixth][first]

\fifth The elements \type {fifth} and \type {sixth} inherit the properties
and counters of \type {first}.

\sixth  Note: inheriting of \type{second} is not allowed because \type
{second} is not an original! \par
```

It may seem very complex but the text below may shed some light on this issue:

first 1

The numbering `first` is unique. We see that one argument is sufficient. By default label and number are placed at the left hand side.

The `second` inherits its counters from `first`, but is placed at the right hand side. In case of three arguments the first one is the copy and the second the original. **second 2**

The numbered elements `third` and `fourth` are both unique and are placed in right margin. **third 1**

Both are defined in one command but they do have own counters that are in no way coupled. **fourth 1**

fifth 3

The elements `fifth` and `sixth` inherit the properties and counters of `first`.

sixth 4

Note: inheriting of `second` is not allowed because `second` is not an original!

It is possible to couple a numbered text element to another. For example we may couple questions and answers. In an interactive document we can click on a question which will result in a jump to the answer. And vice versa. The counters must be synchronised. Be aware of the fact that the counters need some resetting now and then. For example at the beginning of each new chapter. This can be automated by setting the parameter `way` to `bychapter`.

```
\definedescription [question] [coupling=answer]
\definedescription [answer]   [coupling=question]
```

10.4 Indenting

Indented itemizations, like dialogues, can be typeset with the command defined by

```
\defineindenting[...][..,..=..,..]

...      name
..=..    see p 233: \setupindentations
```

After this command `\`*`name`*, `\sub`*`name`* and `\subsub`*`name`* are available.

```
\indentation...\par

...      text
```

The parameters can be set up with the command:

```
\setupindentations[..,...,..][..,..=..,..]
...          name
style        normal bold slanted boldslanted type cap small... command
headstyle    normal bold slanted boldslanted type cap small... command
width        fit dimension
text         text
sample       text
before       command
after        command
distance     dimension
separator    text
```

It is possible to change the setup of `\indentation` with the command:

`\setupindentations[name][setups]`

An example of how you can use the indentation mechanism is given below:

```
\setupindentations
  [sample={rime m},
   separator={:},
   distance=.5em]

\defineindenting[ra][text=rime a]
\defineindenting[rb][text=rime b]
\defineindenting[rc][text=rime c]

\startpacked
\ra  pretty litte girl
\ra  pretty litte girl in a blue dress
\rb  pretty little girl in a blue dress
\rc  playing in the sand
\rb  make my day
\rc  smile for me
\stoppacked
```

This results in:

rime a : pretty litte girl
rime a : pretty litte girl in a blue dress
rime b : pretty little girl in a blue dress
rime c : playing in the sand

rime b : make my day
rime c : smile for me

A series of indenting commands can be enclosed with the commands:

```
\startindentation
\stopindentation
```

10.5 Numbered labels

There is another numbering mechanism that is used for numbering specific text labels that also enables you to refer to these labels. For example, when you want to refer in your text to a number of transparencies that you use in presentations the next command can be used:

```
\definelabel[...][..,..=..,..]

...          name
text         text
location     inmargin intext
way          bytext bysection bychapter
blockway     yes no
headstyle    normal bold slanted boldslanted type cap small... command
headcolor    name
before       command
after        command
```

Where the parameter `location` is set at `intext` and `inmargin`. After this definition the following commands are available:

```
\resetname
\incrementname
\nextname
\currentname[reference]
```

The `[reference]` after `currentname` is optional. After

```
\definelabel[video][text=video,location=inmargin]
```

video 10.1 This defines `\video`, that results in a numbered label *video* in the margin. The command `\currentvideo` would have resulted in the number 0. The label can also be recalled with:

```
\labeling[ref]
```

In our case, saying `\video` results in the marginal note concerning a video. The values of `before` and `after` are executed around the label (which only makes sense for in–text labels.

10.6 Itemize

Items in an itemization are automatically preceded by symbols or by enumerated numbers or characters. The symbols and the enumeration can be set up (see table 10.1). The layout can also be influenced. Itemization has a maximum of four levels.

setup	result	setup	result
n	1, 2, 3, 4	1	dot (•)
a	a, b, c, d	2	dash (–)
A	A, B, C, D	3	star (⋆)
KA	A, B, C, D	4	triangle (▷)
r	i, ii, iii, iv	5	circle (∘)
R	I, II, III, IV	6	big circle (○)
KR	I, II, III, IV	7	bigger circle (◯)
m	1, 2, 3, 4	8	square (□)
g	α, β, γ		
G	Α, Β, Γ		

Table 10.1 Item separator identifications in itemizations.

The command to itemize is:

```
\startitemize[setups]
\item ........
\item ........
\stopitemize
```

So you can do things like this:

```
Which of these theses are true?

\startitemize[A]
\item The difference between a village and a city is the existence of
      a townhall.
```

```
\item The difference between a village and a city is the existence of
      a courthouse.
\stopitemize
```

This will lead to:

Which of these theses are true?

A. The difference between a village and a city is the existence of a townhall.

B. The difference between a village and a city is the existence of a courthouse.

The symbols used under `1` to `8` can be defined with the command `\definesymbol` (see section ??) and the conversion of the numbering with `\defineconversion` (see section ??). For example:

```
Do the following propositions hold some truth?

\definesymbol[1][$\diamond$]

\startitemize[1]
\item The city of Amsterdam is built on wooden poles.
\item The city of Rome was built in one day.
\stopitemize
```

results in:

Do the following propositions hold some truth?

◇ The city of Amsterdam is built on wooden poles.

◇ The city of Rome was built in one day.

The keys `n`, `a`, etc. are related to the conversions. This means that all conversions are accepted. Take for example:

α. a `g` for Greek characters
β. a `G` for Greek capitals

When the setup and the `[ ]` are left out then the default symbol is typeset.

The indentation and horizontal whitespace is set up locally or globally with:

```
\setupitemize[.1.][..,.2.,..][..,..=..,..]
```

.1.	*number* each
.2.	standard *n**broad *n**serried packed unpacked stopper joinedup atmargin inmargin autointro loose section
margin	no standard *dimension*
width	*dimension*
distance	*dimension*
factor	*number*
items	*number*
before	*command*
inbetween	*command*
after	*command*
beforehead	*command*
afterhead	*command*
headstyle	normal bold slanted boldslanted type cap small... *command*
marstyle	normal bold slanted boldslanted type cap small... *command*
symstyle	normal bold slanted boldslanted type cap small... *command*
stopper	*text*
n	*number*
symbol	*number*
align	left right normal
indentnext	yes no

These arguments may appear in different combinations, like:

```
What proposition is true?

\startitemize[a,packed][stopper=:]
\item 2000 is a leap-year
\item 2001 is a leap-year
\item 2002 is a leap-year
\item 2003 is a leap-year
\stopitemize
```

this will become:

What proposition is true?

a: 2000 is a leap-year
b: 2001 is a leap-year
c: 2002 is a leap-year
d: 2003 is a leap-year

Both argument are optional. The key `packed` is one of the most commonly used:

```
What proposition is true?
```

```
\startitemize[n,packed,inmargin]
\item[ok] 2000 is a leap-year
\item 2001 is a leap-year
\item 2002 is a leap-year
\item 2003 is a leap-year
\stopitemize
```

will result in:

What proposition is true?

1. 2000 is a leap-year
2. 2001 is a leap-year
3. 2002 is a leap-year
4. 2003 is a leap-year

It happens very often that an itemization is preceded by a sentence like "*. . . can be seen below:*". In that case we add the key `intro` and the introduction sentence will be 'connected' to the itemization. After this setup a pagebreak between sentence and itemization is discouraged.

```
\startitemize[n,packed,inmargin,intro]
```

The setup of the itemization commands are presented in table 10.2.

In the last example we saw a reference point behind the command `\item` for future cross referencing. In this case we could make a cross reference to answer 1 with the command `\in[ok]`.

The enumeration may be continued by adding the key `continue`, for example:

```
\startitemize[continue]
\item 2005 is a leap-year
\stopitemize
```

This would result in a rather useless addition:

5. 2005 is a leap-year

Another example illustrates that `continue` even works at other levels of itemizations:

setup	**result**
`standard`	default setup
`packed`	no white space between items
`joinedup`	no white space before and after itemization
`paragraph`	no white space before an itemization
n`*serried`	little horizontal white space after symbol
n`*broad`	extra horizontal white space after symbol
`inmargin`	item separator in margin
`atmargin`	item separator at the margin
`stopper`	punctuation after item separator
`intro`	no pagebreak
`columns`	two columns

Table 10.2 Setup of `\setupitemize`.

- **supported image formats in** PDFTEX
  a. png
  b. eps
  c. pdf
- **non supported image formats in** PDFTEX
  d. jpg
  e. gif
  f. tif

This was typed as (in this document we have set `headstyle=bold`):

```
\startitemize[1,packed]
\head  supported image formats in \PDFTEX \par
      \startitemize[a]
      \item png \item eps \item pdf
      \stopitemize
\head  non supported image formats in \PDFTEX \par
      \startitemize[continue]
      \item jpg \item gif \item tif
      \stopitemize
```

```
\stopitemize
```

When we use the key `columns` the items are typeset in two columns. The number of columns can be set by the keys `one`, `two` (default), `three` or `four`.

```
\startitemize[n,columns,four]
\item png \item tif \item jpg \item eps \item pdf
\item gif \item pic \item bmp \item bsd \item jpe
\stopitemize
```

We can see that we can type the items at our own preference.

1. png	4. eps	7. pic	10.jpe
2. tif	5. pdf	8. bmp	
3. jpg	6. gif	9. bsd	

In such a long enumerated list the horizontal space between itemseparator and text may be too small. In that case we use the key `broad`, here `2*broad`:

I. png	IV. eps	VII. pic	X. jpe
II. tif	V. pdf	VIII. bmp	
III. jpg	VI. gif	IX. bsd	

The counterpart of `broad` is `serried`. We can also add a factor. Here we used `2*serried`.

•What format is this?

We can abuse the key `broad` for very simple tables. It takes some guessing to reach the right spacing.

This results in a rather strange example:

```
\startitemize[4*broad,packed]
\sym {yes} this is a nice format
\sym {no}  this is very ugly
\stopitemize
```

yes	this is a nice format
no	this is very ugly

The parameter `stopper` expects a character of your own choice. By default it is set at a period. When no level is specified and the `[ ]` are empty the actual level is activated. In section ?? we will discuss this in more detail. Stoppers only apply to ordered (numbered) list.

There are itemizations where a one line head is followed by a text block. In that case you use `\head` instead of `\item`. You can specify the layout of `\head` with the command `\setupitemize`. For example:

```
\setupitemize[each][headstyle=bold]

\startitemize[n]

\head A title head in an itemization

      After the command \type{\head} an empty line is mandatory. If you
      leave that out you will get a very long header.

\stopitemize
```

This becomes:

1. **A title head in an itemization**

   After the command `\head` an empty line is mandatory. If you leave that out you will get a very long header.

If we would have used `\item` the head would have been typeset in a normal font. Furthermore a pagebreak could have been introduced between head and textblock. This is not permitted when you use `\head`.

```
\head[ref,..]
```

When an itemization consists of only one item we can leave out the commands `\startitemize` and `\stopitemize` and the level 1 symbol is used.

```
\item The itemization commands force the user into a consistent layout
      of the itemizations. \par
```

Instead of the `\par` you could have used an empty line. In each case, we get the following output:

- The itemization commands force the user into a consistent layout of the itemizations.

Only the text directly following the command and ended by an empty line or `\par` is indented.

When you want to re-use the last number instead of increasing the next item you can use `\sub`. This feature is used in discussion documents where earlier versions should not be altered too much for reference purposes.

1. This itemization is preceded by `\startitemize[n,packed]`.
+1. This item is preceded by `\sub`, the other items by `\item`.
2. The itemization is ended by `\stopitemize`.

The most important commands are:

```
\startitemize[..,...,..][..,..=..,..] ... \stopitemize
...      a A KA n N m r R KR number continue standard n*broad n*serried packed stopper joinedup
         atmargin inmargin intro columns
..=..    see p 237: \setupitemize
```

```
\item[ref,..]
```

```
\sub[ref,..]
```

In addition to `\item` there is `\sym`. This command enables us to type an indented text with our own symbol.

```
\sym{...}
```

Another alternative to `\item` is `\mar`. The specified argument is set in the margin (by default a typeletter) and enables us to comment on an item.

```
\mar[ref,..]{...}
```

Some at first sight rather strange alternatives are:

```
\its[ref,..]
```

```
\ran{...}
```

These acronyms are placeholders for `items` and `range`. We illustrate most of these commands with an example that stems from a NTG questionnaire:

no yes
○ ○ ○ ○ ○ I can not do without TEX.
○ ○ ○ ○ ○ I will use TEX forever.
○ ○ ○ ○ ○ I expect an alternative to TEX in the next few years.
○ ○ ○ ○ ○ I use TEX and other packages.
○ ○ ○ ○ ○ I hardly use TEX.
○ ○ ○ ○ ○ I am looking for another system.

The source is typed below. Look at the setup, it is local.

```
\startitemize[5,packed][width=8em,distance=2em,items=5]

\ran {no\hss yes}

\its I can not do without \TeX.
\its I will use \TeX\ forever.
\its I expect an alternative to \TeX\ in the next few years.
\its I use \TeX\ and other packages.
\its I hardly use \TeX.
\its I am looking for another system.

\stopitemize
```

For the interactive version there is:

```
\but[ref]                                              ◄ ►
```

This command resembles `\item` but produces an interactive symbol that executes the reference sequence specified.

The example below shows a combination of the mentioned commands. We also see the alternative `\nop`.

- **he got a head ache**

  1. of all the items
     he had to learn at school

  ++ 2. because the marginal explanation

  +2. of the substantial content

  # turned out to be mostly symbolic

This list was typed like this:

```
\startitemize
\head  he got a head ache

      \startitemize[n,packed]
      \item     of all the items
      \nop      he had to learn at school
      \mar{++}  because the marginal explanation
      \sub      of the substantial content
      \sym{\#}  turned out to be mostly symbolic
      \stopitemize
\stopitemize
```

With the no-operation command:

```
\nop
```

During the processing of itemizations the number of items is counted. This is the case with all versions. The next pass this information is used to determine the optimal location to start a new page. So do not despair when at the first parse your itemizations do not look the way you expected. When using TeXEXEC this is all taken care of.

We have two last pieces of advises. When items consist of two or more paragraphs always use `\head` instead of `\item`, especially when the first paragraph consists only one line. The command `\head` takes care of adequate pagebreaking between two paragraphs. Also, always use the key `[intro]` when a one line sentence preceeds the itemization. This can be automated by:

```
\setupitemize[each][autointro]
```

10.7 Items

A rarely used variant of producing lists is the command `\items`. It is used to produce simple, one level, vertical or horizontal lists. The command in its simplest form looks like this:

`\items{`*`alternative 1`*`,`*`alternative 2`*`,...,`*`alternative N`*`}`

Instead of an alternative you can also type `-`. In that case space is reserved but the item is not set. The layout of such a list is set with the command:

```
\setupitems[..,..=..,..]

location     left right inmargin top bottom
symbol       1 2 ... n a ... text none
width        dimension
n            number unknown
before       command
inbetween    command
align        left right middle margin
after        command
```

The number (n) as well as the width are calculated automatically. When you want to do this yourself you can use the previous command or you pass the options directly. We show some examples.

```
\items[location=left]{png,eps,pdf}
```

∘ png
∘ eps
∘ pdf

```
\items[location=bottom]{png,eps,pdf}
```

png eps pdf

∘ ∘ ∘

```
\items[location=right,width=2cm]{png,eps,pdf}
```

png ∘
eps ∘
pdf ∘

```
\items[location=top,width=6cm,align=left]{png,eps,pdf}
```

∘ ∘ ∘

png eps pdf

```
\items[location=inmargin]{png,eps,pdf}
```

∘ png
∘ eps
∘ pdf

```
\items[location=left,n=2,symbol=5]{jpg,tif}
```

◦ jpg
◦ tif

```
\items[symbol=3,n=6,width=\hsize,location=top]{png,eps,pdf,jpg,tif}
```

⋆	⋆	⋆	⋆	⋆	⋆
	png	eps	pdf	jpg	tif

The setup just after `\items` have the same effect as those of `\setupitems`:

```
\items[..,..=..,..]{..,...,..}
..=..    see p 245: \setupitems
```

10.8 Citations

The use of quotes depends on the language of a country: ‘Nederlands’, ‘English’, ‚Deutsch‘, ‹Français›. The consistent use of single and double quotes is supported by a number of commands. A citation in the running text is typeset by:

```
\startquotation[..,...,..] ... \stopquotation
...      n*left n*middle n*right
```

This command can be compared with `\startnarrower` and has the same setup parameters. The quotes are placed around the text and they fall outside the textblock:

“In commercial advertising ‘experts’ are quoted. Not too long ago I saw a commercial where a washing powder was recommended by the Dutch Society of Housewives. The remarkable thing was that there was a spokesman and not a spokeswoman. He was introduced as the “director”. It can’t be true that the director of the Society of Housewives is a man. Can it?”

In this example we see two other commands:

```
\startquotation
In commercial advertising \quote {experts} are quoted. Not too
long ago I saw a commercial where a washing powder was recommended
by the Dutch Society of Housewives. The remarkable thing was that
there was a spokesman and not a spokeswoman. He was introduced as
```

```
the \quotation {director}. It can’t be true that the director of the
Society of Housewives is a man. Can it?
\stopquotation
```

The command `\quotation` produces double quotes and `\quote` single quotes.

```
\quote{...}
...      text
```

```
\quotation{...}
...      text
```

These commands adapt to the language. In Dutch, English, German and French texts other quotes are activated. The body font is set with:

```
\setupquote[..,..=..,..]
before    command
after     command
style     normal bold slanted boldslanted type cap small... command
```

The location of a period, inside or outside a citation is somewhat arbitrary. The opinions on this issue differ considerately.

He said: “That is a bike” to which she replied: “Take a hike”.

The quotes are language dependent. Therefore it is of some importance that language switching is done correctly.

```
\quotation {He answered: \fr \quotation {Je ne parle pas fran\c cais}.}
\quotation {He answered: \quotation {\fr Je ne parle pas fran\c cais}.}
\quotation {\fr Il respondait: \quotation{Je ne parle pas fran\c cais}.}
\fr \quotation {Il respondait: \quotation{Je ne parle pas fran\c cais}.}
```

Watch the subtle difference.

“He answered: «Je ne parle pas français».”
“He answered: “Je ne parle pas français”.”
“Il respondait: «Je ne parle pas français».”
«Il respondait: «Je ne parle pas français».»

When we want different quotes, we can change them. This is a language related setting.

```
\setuplanguage
  [en]
  [leftquote=\upperleftsinglesixquote,
   leftquotation=\upperleftdoublesixquote]
```

Fo rconsistency, such a setting can best be put into the local system file `cont-sys.tex`, together with other local settings. The following quotes are available:

`\lowerleftsingleninequote`	‚	`\lowerrightsingleninequote`	‚
`\lowerleftdoubleninequote`	„	`\lowerrightdoubleninequote`	„
`\upperleftsingleninequote`	’	`\upperrightsingleninequote`	’
`\upperleftdoubleninequote`	”	`\upperrightdoubleninequote`	”
`\upperleftsinglesixquote`	‘	`\upperrightsinglesixquote`	‘
`\upperleftdoublesixquote`	“	`\upperrightdoublesixquote`	“

Lines and frames

11

11.1 Introduction

TEX has an enormous capacity in handling text, but is very weak at handling graphical information. Lines can be handled adequately as long as you use vertical or horizontal lines. However, you can do graphical work with TEX by combining TEX and METAPOST.

In this chapter we introduce a number of commands that relate to drawing straight lines in your text. We will see a very sophisticated command `\framed` that can be used in many ways. The parameters of this command are also available in other commands.

11.2 Single lines

The simplest way to draw a line in ConTEXt is:

```
\hairline
```

For example:

```
\hairline
In what fairy tale is the wolf cut open and filled with stones? Was it in
{Little Red Riding-hood} or in \quote {The wolf and the seven goats}.
\hairline
```

This will become:

In what fairy tale is the wolf cut open and filled with stones? Was it in Little Red Riding-hood or in ‘The wolf and the seven goats’.

It does not look good at all. This is caused by the fact that a drawn line gets its own vertical whitespace. In section 11.4 we will show how to alter this.

The effects of the command `\hairline` is best illustrated when we visualize `\strut`’s. We did so by saying `\showstruts` first.

A strut is a character with a maximum height and depth, but no width. The text in this example is surrounded by two strutted lines.

It is also possible to draw a line over the width of the actual paragraph:

```
\thinrule
```

Or more than one lines by:

```
\thinrules[..=..]
..=..    see p 253: \setupthinrules
```

For example:

```
\startitemize
\item question 1 \par \thinrules[n=2]
\item question 2 \par \thinrules[n=2]
\stopitemize
```

If you leave out a `\par` (or empty line), the thin rules come after the text. Compare

- question 1
- question 2

with

- question 1
- question 2

The last example was keyed in as:

```
\startitemize
\item question 1 \thinrules[n=2]
\item question 2 \thinrules[n=2]
\stopitemize
```

The parameters are set with:

```
\setupthinrules[..=..]
interlinespace    small medium big
n                 number
before            command
after             command
```

You can draw thin vertical or horizontal lines with the commands:

```
\vl[...]
...      number
```

```
\hl[...]
...      number
```

The argument is optional. To `\vl` (|) you may pass a factor that relates to the actual height of a line and to `\hl` (__) a width that relates to the width of an em. So `\vl[2]` produces a rule with a height of two lines.

11.3 Fill in rules

On behalf of questionnaires there is the command:

```
\fillinline[..,..=..,..]...\par
..=..    see p 254: \setupfillinlines
```

With the accompanying setup command:

```
\setupfillinlines[..,..=..,..]
width       dimension
margin      dimension
distance    dimension
before      command
after       command
```

The example:

```
\fillinline[n=2,width=2cm]{name} \par
\fillinline[n=2,width=2cm]{address} \par
```

Leads to the next list:

name __________

address __________

An alternative is wanting the fill–in rule at the end of a paragraph. Then you use the commands:

```
\fillinrules[..,..=..,..]{.1.}{.2.}
..=..    see p 254: \setupfillinrules
```

```
\setupfillinrules[..,..=..,..]
width              fit broad dimension
distance           dimension
before             command
after              command
style              normal bold slanted boldslanted type cap small... command
n                  number
interlinespace     small medium big
separator          text
```

The next example will show the implications:

```
\fillinline[width=3cm] Consumers in this shopping mall are frequently
confronted with questionnaires. Our hypothesis is that consumers rather
shop somewhere else than answer these kind of questionnaires. Do you
agree with this?
```

In this example we could of course have offered some alternatives for answering this question. By setting the width to broad, we get

Consumers in this shopping mall are frequently confronted with questionnaires. Our hypothesis is that consumers rather shop somewhere else than answer these kind of questionnaires. Do you agree with this? ______________

The next set of examples demonstrate how we can influence the layout.

```
\fillinrules[n=2,width=fit]{first}
\fillinrules[n=2,width=broad]{first}
\fillinrules[n=2,width=3cm]{first}
\fillinrules[n=2,width=fit,distance=.5em,separator=:]{first}
\fillinrules[n=2,width=broad,distance=.5em]{first}{last}
```

first____

____

first ____

____

first   ________________________________

____

first:____

____

first ____

____ last

11.4 Text lines

A text line is drawn just before and/or after a paragraph. The upper line may also contain text. The command is:

```
\textrule[.1.]{.2.}
.1.     top bottom
.2.     text
```

An example:

```
\textrule[top]{Instruments}
Some artists mention the instruments that they use during the production
of their \kap{CD}. In Peter Gabriel’s \quote {Digging in the dust} he used
the {\em diembe}, {\em tama} and {\em surdu}. The information on another
song mentions the {\em doudouk}. Other \quote {unknown} instruments are
used on his \kap{cd} \quote {Passion}.
\textrule
```

This will result in:

—— **Instruments** ————————————————————

Some artists mention the instruments that they use during the production of their CD. In Peter Gabriel’s ‘Digging in the dust’ he used the *diembe*, *tama* and *surdu*. The information on another song mentions the *doudouk*. Other ‘unknown’ instruments are used on his CD ‘Passion’.

The behaviour of textlines is set up with the command below. With the parameter `width` you set the length of the line in front of the text.

```
\setuptextrules[..,..=..,..]
```

location	left inmargin
before	*command*
after	*command*
inbetween	*command*
width	*dimension*
distance	*dimension*
bodyfont	5pt ... 12pt small big
color	*name*
style	normal bold slanted boldslanted type cap small... *command*
rulecolor	*name*

These is also a `\start-\stop` alternative. This one also honors the `bodyfont` parameter.

```
\starttextrule[.1.]{.2.} ... \stoptextrule
```

.1.	top bottom
.2.	*text*

11.5 Underline

Underlining text is not such an ideal method to banner your text. Nevertheless we introduced this feature in ConTeXt. Here is how it works. We use:

```
\underbar{...}
...     text
```

A disadvantage of this command is that words can no longer be hyphenated. This is a nasty side–effect. But we do support nested underlining.
The spaces in the last paragraph were also underlined. If we do not want that in this paragraph we use:

```
\underbars{..  ... ..}
...     text
```

From the input we can see that the hyphen results from the compound word.

```
\underbar {A disadvantage of this command is that words can \underbar
{no} longer be hyphenated. This is a nasty side||effect. But we do
support \underbar {nested} underlining.}
```

```
\underbars {The spaces in the last paragraph were also underlined. If
we do not want that in this paragraph we use:}
```

The counterpart of these commands are:

```
\overbar{...}
...     text
```

```
\overbars{..  ... ..}
...     text
```

You may wonder for what reasons we introduced these commands. The reasons are mainly financial:

product 1	1.420
product 2	3.182
total	4.602

This financial overview is made with:

```
\starttabulate[|l|r|]
\NC product 1 \NC          1.420  \NC \NR
\NC product 2 \NC          3.182  \NC \NR
\NC total     \NC \overbar{4.602} \NC \NR
\stoptabulate
```

The number of parameters in these commands is limited:

```
\setupunderbar[..,..=..,..]

alternative     a b c
rulethickness   dimension
```

The alternatives are: alternative a, alternative b, alternative c while another line thickness results in: 1pt line, 2pt line.

A part of the text can be ~~striked~~ with the command:

```
\overstrike{...}

...     text
```

This command supports no nesting. Single ~~words~~ ~~are~~ ~~striked~~ with:

```
\overstrikes{..  ... ..}

...     text
```

11.6 Framing

Texts can be framed with the command: `\framed`. In its most simple form the command looks like this:

```
\framed{A button in an interactive document is a framed text
with specific characteristics.}
```

The becomes:

A button in an interactive document is a framed text with specific characteristics.

The complete definition of this command is:

```
\framed[..,..=..,..]{...}
..=..    see p 264: \setupframed
...      text
```

You may notice that all arguments are optional.

```
\framed
  [height=broad]
  {A framed text always needs special attention as far as the spacing
   is concerned.}
```

Here is the output of the previous source code:

A framed text always needs special attention as far as the spacing is concerned.

For the height, the values `fit` and `broad` have the same results. So:

```
\hbox
  {\framed[height=broad]{Is this the spacing we want?}
   \hskip1em
   \framed[height=fit]  {Or isn't it?}}
```

will give us:

Is this the spacing we want? Or isn't it?

To obtain a comparable layout between framed and non-framed framing can be set on and off.

yes	no	yes
no	yes	no

The rulethickness is set with the command `\setuprulethickness` (see section ??).

A framed text is typeset 'on top of' the baseline. When you want real alignment you can use the command `\inframed`.

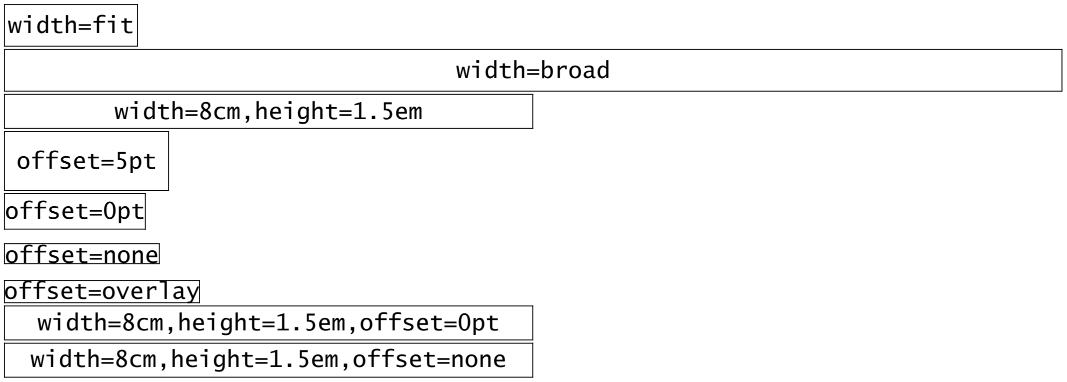
```
to \framed{frame} or to be \inframed{framed}
```

or:

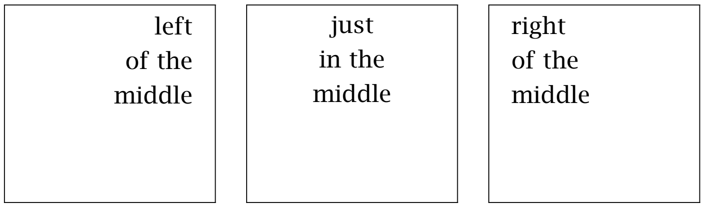
to $\boxed{\text{frame}}$ or to be $\boxed{\text{framed}}$

It is possible to draw parts of the frame. In that case you have to specify the separate sides of the frame with `leftframe=on` and the alike.

We will now show some alternatives of the command `\framed`. Please notice the influence of `offset`. When no value is given, the offset is determined by the height and depth of the `\strut`, that virtual character with a maximum height and depth with no width. When exact positioning is needed within a frame you set `offset` at `none` (see also tables 11.1, 11.2 and 11.3). Setting the `offset` to `none` or `overlay`, will also disable the strut.



The commands `\lbox` (ragged left), `\cbox` (ragged center) and `\rbox` (ragged right) can be combined with `\framed`:



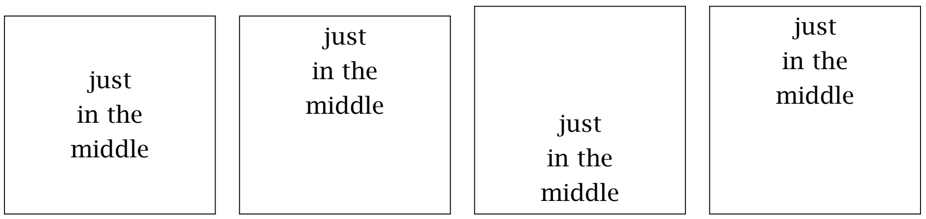
`\lbox` `\cbox` `\rbox`

The second text is typed as follows:

```
\framed
  [width=.2\hsize,height=3cm]
  {\cbox to 2.5cm{\hsize2.5cm just\\in the\\middle}}
```

There is a more convenient way to align a text, since we have the parameters `align` and `top` and `bottom`. In the next one shows the influence of `top` and `bottom` (the second case is the default).

```
\setupframed[width=.2\hsize,height=3cm,align=middle]
\startcombination[4]
  {\framed[bottom=\vss,top=\vss]{just\\in the\\middle}}
  {\type{top=\vss}\crlf\type{bottom=\vss}}
  {\framed[bottom=\vss,top=]    {just\\in the\\middle}}
  {\type{top=}    \crlf\type{bottom=\vss}}
  {\framed[bottom=,top=\vss]    {just\\in the\\middle}}
  {\type{top=\vss}\crlf\type{top=}}
  {\framed[bottom=,top=]        {just\\in the\\middle}}
  {\type{top=}    \crlf\type{bottom=}}
\stopcombination
```



top=\vss
bottom=\vss
	top=
	bottom=\vss
		top=\vss
		top=
			top=
			bottom=

In the background of a framed text you can place a screen or a coloured background by setting `background` at `color` or `screen`. Don't forget to activate the the colour mechanism by saying (`\setupcolors[state=start]`).

`background=screen`

`background=screen`
`backgroundscreen=0.7`

`background=color`

`background=color`
`backgroundcolor=red`

There is also an option to enlarge a frame or the background by setting the `frameoffset` and/ or `backgroundoffset`. These do not influence the dimensions. Next to screens and colours you can also use your own kind of backgrounds. This mechanism is described in section 6.7.

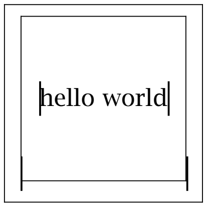
The command `\framed` itself can be an argument of `\framed`. We will obtain a framed frame.

```
\framed
  [width=3cm,height=3cm]
  {\framed[width=2.5cm,height=2.5cm]{hello world}}
```

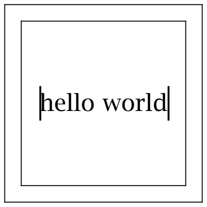
In that case the second frame is somewhat larger than expected. This is caused by the fact that the first framed has a strut. This strut is placed automatically to enable typesetting one framed text next to another. We suppress `\strut` with:

```
\framed
  [width=3cm,height=3cm,strut=no]
  {\framed[width=2.5cm,height=2.5cm]{hello world}}
```

When both examples are placed close to one another we see the difference:



`strut=yes`



`strut=no`

A `\hairline` is normally draw over the complete width of a text (`\hsize`). Within a frame the line is drawn from the left to the right of framed box.

Consequently the code:

```
\framed[width=8cm,align=middle]
  {when you read between the lines \hairline
   you may see what effort it takes \hairline
   to write a macropackage}
```

produces the following output:

when you read between the lines
you may see what effort it takes
to write a macropackage

When no width is specified only the vertical lines are displayed.

their opinions	differ	considerately

Which was obtained with:

```
\framed
  {their opinions \hairline differ \hairline considerately}
```

The default setup of `\framed` can be changed with the command:

```
\setupframed[..,..=..,..]
```

height	fit broad *dimension*
width	fit broad *dimension*
offset	none overlay *dimension*
location	low depth
option	none empty
strut	yes no
align	no left right middle normal
bottom	*command*
top	*command*
frame	on off overlay
topframe	on off
bottomframe	on off
leftframe	on off
rightframe	on off
frameoffset	*dimension*
framedepth	*dimension*
framecorner	round rectangular
frameradius	*dimension*
framecolor	*name*
background	screen color none foreground *name*
backgroundscreen	*number*
backgroundcolor	*name*
backgroundoffset	*dimension*
backgrounddepth	*dimension*
backgroundcorner	round rectangular
backgroundradius	*dimension*
depth	*dimension*
corner	round rectangular
radius	*dimension*
empty	yes no
...	*text*

The command `\framed` is used within many other commands. The combined use of `offset` and `strut` may be very confusing. It realy pays off to spend some time playing with these macros and parameters, since you will meet `\framed` in many other commands. Also, the parameters `width` and `height` are very important for the framing texts. For that reason we summarize the consequences of their settings in table 11.1, 11.2 and 11.3.

		offset			
		.25ex	0pt	none	overlay
strut	yes	▯	│	.	
	no	▫	.	.	

Table 11.1 The influence of `strut` and `offset` in `\framed` (1).

		offset			
		.25ex	0pt	none	overlay
strut	yes	$\boxed{\TeX}$	$\boxed{\TeX}$	$\boxed{\TeX}$	$\boxed{\TeX}$
	no	$\boxed{\TeX}$	$\boxed{\TeX}$	$\boxed{\TeX}$	$\boxed{\TeX}$

Table 11.2 The influence of `strut` and `offset` in `\framed` (2).

		width	
		fit	broad (\hsize=4cm)
height	fit	$\boxed{\text{xxxx}}$	$\boxed{\quad\quad\text{xxxx}\quad\quad}$
	broad	$\boxed{\text{xxxx}}$	$\boxed{\quad\quad\text{xxxx}\quad\quad}$

Table 11.3 The influence of `height` and `width` in `\framed`.

happy
birthday
to you

At first sight it is not so obvious that `\framed` can determine the width of a paragraph by itself. When we set the parameter `align` the paragraph is first typeset and then framed. This feature valuable when typesetting titlepages. In the example left of this text, linebreaks are forced by \\, but this is not mandatory. This example was coded as follows:

```
\placefigure
  [left]
  {none}
  {\framed[align=middle]{happy\\birthday\\to you}}
```

The parameter `offset` needs some special attention. By default it is set at `.25ex`, based on the cureently selected font. The next examples will illustrate this:

```
\hbox{\bf \framed{test} \sl \framed{test} \tfa \framed{test}}
\hbox{\framed{\bf test} \framed{\sl test} \framed{\tfa test}}
```

The value of `1ex` outside `\framed` determines the offset. This suits our purpose well.

test *test* test
test *test* test

The differences are very subtle. The distance between the framed boxes depends on the actual font size, the dimensions of the frame, the offset, and the strut.

TEX can only draw straight lines. Curves are drawn with small line pieces and effects the size of DVI–files considerately and will cause long processing times. Curves in CONTEXT are implemented by means of POSTSCRIPT. There are two parameters that affect curves: `corner` and `radius`. When `corner` is set at `round`, round curves are drawn.

Don't be to edgy.

It is also possible to draw circles by setting `radius` at half the width or height. But do not use this command for drawing, it is meant for framing text. Use METAPOST instead.

Technically speaking the background, the frame and the text are separate components of a framed text. First the background is set, then the text and at the last instance the frame. The curved corner of a frame belongs to the frame and is not influenced by the text. As long as the radius is smaller than the offset no problems will occur.

11.7 Framed texts

When you feel the urge to put a frame around or a backgroud behind a paragraph there is the command:

```
\startframedtext[...][..,..=..,..] ... \stopframedtext
...      left right middle
..=..    see p 267: \setupframedtexts
```

An application may look like this:

```
\startframedtext[left]
From an experiment that was conducted by C. van Noort (1993) it was
shown that the use of intermezzos as an attention enhancer is not very
effective.
\stopframedtext
```

From an experiment that was conducted by C. van
Noort (1993) it was shown that the use of intermezzos
as an attention enhancer is not very effective.

This can be set up with:

```
\setupframedtexts[..,..=..,..]

bodyfont    5pt ... 12pt small big
style       normal bold slanted boldslanted type small... command
left        command
right       command
before      command
after       command
inner       command
margin      standard yes no
..=..       see p 264: \setupframed
```

Framed texts can be combined with the place block mechanism, as can be seen in intermezzo 11.1.

```
\placeintermezzo
  [here][int:demo 1]
  {An example of an intermezzo.}
  \startframedtext
    For millions of years mankind lived just like animals. Then
    something happened, which unleashed the power of our imagination.
    We learned to talk.
    \blank
    \rightaligned{--- The Division Bell / Pink Floyd}
  \stopframedtext
```

In this case the location of the framed text (between `[ ]`) is left out.

For millions of years mankind lived just like animals.
Then something happened, which unleashed the
power of our imagination. We learned to talk.

— The Division Bell / Pink Floyd

Intermezzo 11.1 An example of an intermezzo.

You can also draw a partial frame. The following setup produces intermezzo 11.2.

```
\setupframedtexts[frame=off,topframe=on,leftframe=on]
```

Why are the world leaders not moved by songs
like *Wozu sind Kriege da?* by Udo Lindenberg. I
was, and now I wonder why wars go on and on.

Intermezzo 11.2 An example of an intermezzo.

You can also use a background. When the background is active it looks better to omit the frame.

An intermezzo like this will draw more attention,
but the readability is far from optimal. However,
you read can it. This inermezzo was set up with :

```
\setupframedtexts[frame=off,background=screen]
```

Intermezzo 11.3 An example of an intermezzo with background.

Intermezzo 11.4 demonstrate how to use some color:

```
\setupframedtexts
  [background=screen,
   frame=off,
   rightframe=on,
   framecolor=darkgreen,
   rulethickness=3pt]
```

```
\placeintermezzo
  [here][int:color]
  {An example of an intermezzo with a trick.}
  \startframedtext
    The trick is really very simple. But the fun is gone when Tom, Dick
    and Harry would use it too.
  \stopframedtext
```

The trick is really very simple. But the fun is
gone when Tom, Dick and Harry would use it too.

Intermezzo 11.4 An example of an intermezzo with a trick.

So, in order to get a partial frame, we have to set the whole `frame` to `off`. This is an example of a situation where we can get a bit more readable source when we say:

```
\startbuffer
\startframedtext ... \stopframedtext
\stopbuffer
```

```
\placeintermezzo
  [here][int:color]
  {An example of an intermezzo with a trick.}{\getbuffer}
```

You do not want to set up a framed text every time you need it, so there is the following command:

```
\defineframedtext[...][..,..=..,..]

...      name
..=..    see p 267: \setupframedtexts
```

The definition:

```
\defineframedtext
  [musicfragment]
  [frame=off, rightframe=on, leftframe=on]
```

```
\placeintermezzo
  [here][]
  {An example of a predefined framed text.}
\startmusicfragment
Imagine that there are fragments of music in your interactive document.
You will not be able to read undisturbed.
\stopmusicfragment
```

results in:

Imagine that there are fragments of music in your interactive document. You will not be able to read undisturbed.

Intermezzo 11.5 An example of a predefined framed text.

11.8 Margin rules

To add some sort of flags to paragraphs you can draw vertical lines in the margin. This can be used to indicate that the paragraph was altered since the last version. The commands are:

```
\startmarginrule[...] ... \stopmarginrule
...      number
```

```
\marginrule[.1.]{.2.}
.1.      number
```

The first command is used around paragraphs, the second within a paragraph.

By specifying a level you can suppress a margin rule. This is done by setting the ‘global’ level higher than the ‘local’ level.

```
\setupmarginrules[..=..]
level        number
thickness    dimension
```

In the example below we show an application of the use of margin rules.

```
\startmarginrule
The sound of a duck is a good demonstration of how different people
listen to a sound. Everywhere in Europe the sound is equal. But in
every country it is described differently: kwaak||kwaak (Netherlands),
couin||couin (French), gick||gack (German), rap||rap (Danish) and
mech||mech  (Spanish). If you speak these words aloud you will notice
that \marginrule[4]{in spite of the} consonants the sound is really very
well described. And what about a cow, does it say boe, mboe or mmmmmm?
\stopmarginrule
```

Or:[22]

The sound of a duck is a good demonstration of how different people listen to a sound. Everywhere in Europe the sound is equal. But in every country it is described differently: kwaak–kwaak (Netherlands), couin–couin (French), gick–gack (German), rap–rap (Danish) and mech–mech (Spanish). If you speak these words aloud you will notice that in spite of the consonants the sound is really very well described. And what about a cow, does it say boe, mboe or mmmmmm?

If we would have set `\setupmarginrules[level=2]` we would have obtained a margin rule in the middle of the paragraph. In this example we also see that the thickness of the line is adapted to the level. You can undo this feature with `\setupmarginrules[thickness=1]`.

11.9 Black rules

Little black boxes —we call them black rules— (■) can be drawn by `\blackrule`:

```
\blackrule[..,..=..,..]
..=..    see p 272: \setupblackrules
```

When the setup is left out, the default setup is used.

[22] G.C. Molewijk, Spellingsverandering van zin naar onzin (1992).

```
\setupblackrules[..,..=..,..]
width          dimension max
height         dimension max
depth          dimension max
alternative    a b
distance       dimension
n              number
```

The height, depth and width of a black rule are in accordance with the usual height, depth and width of TEX. When we use the key `max` instead of a real value the dimensions of TEX's `\strutbox` are used. When we set all three dimensions to `max` we get: █.

▬ Black rules may have different purposes. You can use them as identifiers of sections or subsections. This paragraph is tagged by a black rule with default dimensions: `\inleft{\blackrule}`.

A series of black rules can be typeset by `\blackrules`:

```
\blackrules[..,..=..,..]
..=..    see p 272: \setupblackrules
```

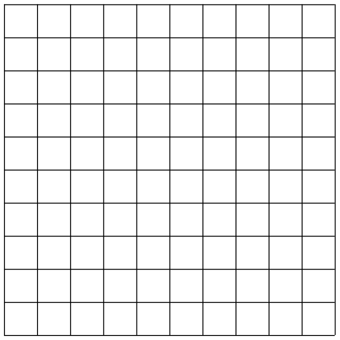
▬▬▬ There are two versions. Version `a` sets `n` black rules next to each other with an equal specified width. Version `b` divides the specified width over the number of rules. This paragraph is tagged with `\inleft{\blackrules}`. The setup after `\blackrule` and `\blackrules` are optional.

11.10 Grids

We can make squared paper (a sort of grid) with the command:

```
\grid[..,..=..,..]
x           number
y           number
nx          number
ny          number
dx          number
dy          number
xstep       number
ystep       number
offset      yes no
factor      number
scale       number
unit        cm pt em mm ex sp in
location    left middle
```

The default setup produces:



It is used in the background when defining interactive areas in a figure. And for the sake of completeness it is described in this chapter.

11 Grids

Blocks

12

```
References

> floats
> stp:380
> stp:definefloat

Registers

> i figures+placing
> i tables+placing
> i figures+numbering
> i tables+numbering
> i figures+listing
> i tables+listing
> i placing+figures
> i placing+tables
> i numbering+figures
> i numbering+tables
> i listing+figures
> i listing+tables
> m \tex {definefloat}
> m \tex {place«float»}
> m \tex {placelistof<..
> m \tex {completelist..
> m \tex {reserve<<flo..
> m \tex {setup«floats»}
> m \tex {start<<float..
> m \tex {setupfloats}
> m \tex {setupcaptions}
> t \tex {definefloat}
```

12.1 Introduction

A block in ConTeXt is defined as typographical unit that needs specific handling. We distinguish the following block types:

- **floats**

  Examples of floats are figures, tables, graphics, intermezzos etc. The locations of these blocks are determined by TeX and depends on the available space on a page.

- **textblocks**

  Examples of textblocks are questions and answers in a studybook, summaries, definitions or derivatives of formulas. The location of these kind of blocks in the final document cannot be determined beforehand. And the information may be used repeatedly in several settings.

- **opposite blocks**

  Opposite (or spread) blocks are typeset on the left–hand page when a single sided output is generated. The layout of the right–hand side page is influenced by the blocks on the left.

- **margin blocks**

  Margin blocks are more extensive than single margin words. Text and figures can be placed in the margin with this feature.

There are a number of commands that support the use of these block types. These are discussed in this chapter. Furthermore we will discuss other forms of text manipulation. Formulas can also be seen as blocks. Since formulas are covered in a separate chapter we don't go into details here.

This chapter is typeset with the option `\version [temporary]`. This does not refer to the content but to the typesetting. With this option, design information is placed in the margin.

12.2 Floats

Floats are composed of very specific commands. For example a table in ConTeXt is typeset using a shell around TaBlE. Drawings and graphics are made with external packages, as TeX is only capable of reserving space for graphics.

Most floats are numbered and may have a caption. A float is defined with the command:

```
\definefloat[.1.][.2.]
.1.      singular name
.2.      plural name
```

In CONTEXT, figures, graphics, tables, and intermezzos are predefined with:

```
\definefloat [figure]     [figures]
\definefloat [table]      [tables]
\definefloat [graphic]    [graphics]
\definefloat [intermezzo] [intermezzos]
```

As a result of these definitions you can always use `\placefigure`, `\placetable`, `\placegraphic` and `\placeintermezzo`. Of course, you can define your own floats with `\definefloat`. You place your newly defined floats with the command:

```
\placefloat[.1.][ref,..]{.2.}{.3.}
.1.      left right here top bottom inleft inright inmargin margin page opposite always force
         tall
.2.      text
.3.      text
```

When a float cannot be placed at a specific location on a page, CONTEXT will search for the most optimal alternative. CONTEXT provides a number of placement options for floats. These are listed in table 12.1.

The commands can be used without the left and right brackets. For example:

```
\place...{caption}{content}
```

When the caption is left out, the float number is generated anyway. When the number is not needed you type `none`, like in:

```
\placefigure{none}{.....}
```

It is mandatory to end this command by an empty line or a `\par`. You don't have to embed a table in braces, since the `\start` and `\stop` commands have them built in:

```
\placetable
  [here][tab:example]
```

Floatblocks

> 83 saved

References

> stp:378
> stp:place@@float@@
< tab:floats
> tab:floats

Registers

> t \tex {place«float»}

preference	result
`left`	left of text
`right`	right of text
`here`	preferably here
`top`	at top of page
`bottom`	at bottom of page
`inleft`	in left margin
`inright`	in right margin
`inmargin`	in the margin (left or right)
`margin`	in the margin (margin float)
`page`	on a new (empty) page
`opposite`	on the left page
`always`	precedence over stored floats
`force`	per se here

83 **Table 12.1** Preferences in `\place`*`float`*.

```
{A very simple example of a table.}
\starttable[|c|c|]
\HL
\VL this \VL is     \VL\FR
\VL a    \VL table  \VL\LR
\HL
\stoptable
```

this	is
a	table

84 **Table 12.2** A very simple example of a table.

The vertical whitespace for a float can be reserved with:

```
\reservefloat[..,..=..,..][.1.][ref,..]{.2.}
height   dimension
width    dimension
frame    on off
.1.      left right here top bottom inleft inright inmargin margin page opposite always force
.2.      text
```

Floatblocks

< 85 placed

References

< fig:reservation
> fig:reservation
> stp:382
> stp:start@@float@@text

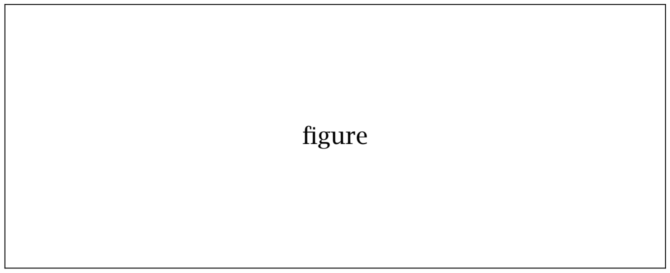
Registers

> t \tex {start<<float..

This command can be used without the left and right bracket. An example of a reservation is:

```
\reservefigure
  [height=4cm,width=10cm,frame=on][here][fig:reservation]
  {An example of a reservation.}
```

Which results in figure 12.1.



85 **Figure 12.1** An example of a reservation.

When the content of a float is not yet available, you can type `\empty...` instead of `\place...`. In this way you can also reserve vertical whitespace. When no option is added, so `{ }` is typed, the default empty float is used. However, whether the figure or table is available is not that important. You can always type:

```
\placefigure{This is a figure caption.}{}
```

As a first argument you can specify a key `left` or `right` that will cause ConTeXt to let the text flow around the float. The second optional parameter can be a cross reference, to be used later, like `\at {page} [fig:schematic process]`.

```
\placefigure[here][fig:demo]{This a figure caption.}{}
```

As we will later see, you can also use the next command:

```
\startfloattext[.1.][ref]{.2.}{.3.} ... \stopfloat
.1.      left right high middle low offset tall
.2.      text
.3.      text
.4.      text
```

References

> stp:383
> stp:placelistof@@flo..
< numbering
> stp:384
> stp:completelistof@@..
< toc
> stp:381
> stp:setupfloat
< \currentsetupinherit..

Registers

> t \tex {placelistof<..
> t \tex {completelist..
> t \tex {setupfloat}

Preferences are `left`, `right` or `middle`. Furthermore you can specify `offset` in case the text should align with the float. Both setups can be combined: `[left,offset]`.

A list of used floats is generated with the command:

```
\placelistoffloats                                          ◀ ▶
```

For example, the command `\placelistoffigures` would typeset a list of figures. The list follows the numbering convention that is set with the command `\setupnumbering`, which was discussed at page ??.

numbering->

The next command generates a list of floats on a separate page.

```
\completelistoffloats                                       ◀ ▶
```

Pagebreaks that occur at unwanted locations can be enforced in the same way that is done with a table of contents (see section 9.1):

```
\completelistoffloats[pageboundaries={8.2,20.4}]
```

As with tables of content the default local lists are generated. Recalling a list within a chapter produces a list for that specific chapter. So, if you want a list of all figures, you need to specify `criterium` as `all`.

The previous list was produced by saying:

```
\placelistoffigures[criterium=chapter]
```

The characteristics of a specific class of floats are specified with the command:

```
\setupfloat[...][..,..=..,..]
...               name
height            dimension
width             dimension
pageboundaries    list
..=..             see p 264: \setupframed
```

References

> stp:237
> stp:setupfloats
< \currentsetupinherit..
> stp:235
> stp:setupcaptions

Registers

> t \tex {setupfloats}
> t \tex {setupcaptions}

The (predefined) floats can also be set up with the more meaningful commands `\setupfigures`, `\setuptables` etc.

The height and width relate to the vertical whitespace that should be reserved for an empty float. All settings of `\framed` can be used, so when `frame` is set to `on`, we get a framed float.

The next two commands relate to *all* floats. The first command is used for setting the layout including the caption:

```
\setupfloats[..,..=..,..]
location          left right middle
width             fit dimension
before            command
after             command
margin            dimension
spacebefore       n*small n*medium n*big none
spaceafter        n*small n*medium n*big none
sidespacebefore   n*small n*medium n*big none
sidespaceafter    n*small n*medium n*big none
ntop              number
nbottom           number
nlines            number
..=..             see p 264: \setupframed
```

The second command is used for setting the enumerated captions of figures, tables, intermezzos, etc.

```
\setupcaptions[..,..=..,..]
```

location	top bottom none high low middle
width	fit max *dimension*
headstyle	normal bold slanted boldslanted type cap small... *command*
style	normal bold slanted boldslanted type cap small... *command*
number	yes no
inbetween	*command*
align	left middle right no
conversion	numbers characters Characters romannumerals Romannumerals
way	bytext by*section*

Floatblocks

< 86 placed

References

> stp:236
> stp:setupcaption
< \currentsetupinherit..

Registers

> t \tex {setupcaption}

You can also set up captions for a specific class of floats, like figures. The first argument of the next command is the name of that class of floats.

```
\setupcaption[...][..,..=..,..]
```

...	*name*
..=..	see p 281: \setupcaptions

The commands assigned to `before`, `after` are are executed before and after placing the float. The parameter `inbetween` is executed between the float and the caption. All three normally have a `\blanko` command assigned.

The parameter `style` is used for numbering (**Figure x.y**) and `width` for the width of the caption label. The parameter `margin` specifies the margin space around a float when it is surrounded by text. The float macros optimize the width of the caption (at `top` or `bottom`) related to the width of the figure or table.

86
```
\setupcaptions[location=high]
\setupfloats[location=left]
```

Figure 12.2

With the three variables `ntop`, `nbottom` and `nlines` the float storage mechanism can be influenced. The first two variables specify the maximum number of floats that are saved per page at the top or the bottom of a page.

By default these variables have the values 2 and 0. Assume that ten figures, tables and/or other floats are stored, then by default two floats will be placed at each new page (if possible).

Floatblocks

< 87 placed
< 88 placed

For example, at a forced pagebreak or at the beginning of a new chapter, all stored floats are placed.

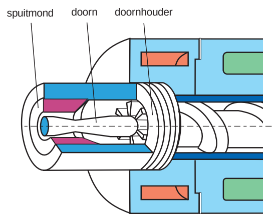
The parameter `nlines` has the default value 4. This means that never less than four lines will be typeset on the page where the floats are placed.

We continue with a few examples of floats (figures) placed next to the running text. This looks like:

```
\placefigure[right]{none}{}
... here is where the text starts ....
```

For illustrating the mechanism we do need some text. Therefore the examples are used to explain some issues on the float mechanism.

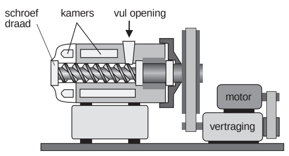
Floats are placed automatically. The order of appearance follows the order you have keyed in the source. This means that larger floats are placed somewhere else in your document. When `\version[temporary]` is set, you can get information on the float mechanism. By consulting that information you get some insight into the process.



Floats can be surrounded by text. The float at the right was set with `\placefigureright[right]{none}{...}`. The float mechanism works automatically. Should it occur that pages are left blank as a result of poor float placement, you will need to make some adaptations manually. You can downsize your figure or table or alter your text. It is also a good practice to define your float some paragraphs up in your source. However, all of this should be done at the final 87 production stage.

With the key `force` you can force a float to be placed at that exact location. Tables or figures that are preceded by text like: 'as we can see in the figure below' may be defined with this option.

88

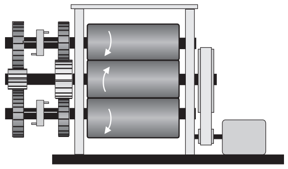


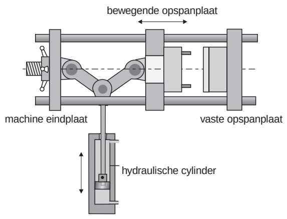
In manuals and study books we encounter many illustrations. It is almost unavoidable to manually adapt these for optimal display. However, the float commands in ConTEXt are optimized in such a way that you can produce books with hundreds of floats effortlessly. The worst case is that some floats are stored and placed at the end of the chapter. But this can be influenced with the command `\startpostponing`. Postponing is done with the keys `always` which can be combined with the location, like `[left,always]` or `[here,always]`. Because the order of the floats is changed several parses are necessary for the document. These processes can be traced via messages on the terminal.

This brings us to a figure that is placed at the left side of a page. The side float mechanism in inspired and based on a mechanism of D. Comenetz. In the background three mechanisms are active. A mechanism to typeset a figure on top, inbetween, of under existing text. There is a mechanism to place figures on the right or left of a page. And there is a third mechanism to typeset text next to a figure.

We see an example of the last mechanism. The text is enclosed by the commands:

```
\startfiguretext
   [right]{none}{\externalfigure[rb00015]}
....
\stopfiguretext
```


89



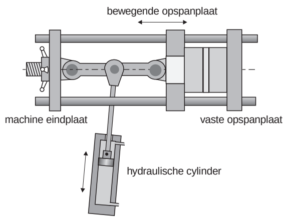
90

It is obvious that we can also place the figure at the left. With `\start...text` we can add the key `offset`. Here we used `[left,offset]`.

When the text is longer than expected, then it will *not* flow around the float. By default the floats are handled in the same order they are typed in the source file. This means that the stored figures are placed first. If this is not desired you can type the key `always`. The actual float will get priority.

There are more options. In this case the setup `[right,middle]` is given. In the same way we place text `high` and `low`.

When the key `long` is used the rest of the text is filled out with empty lines, as here.



91

When several figures are set under each other, making them the same width makes for a nice presentation on the page. This looks better.

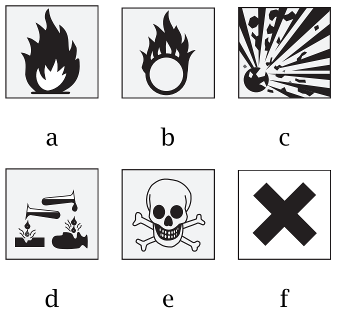
```
Floatblocks

< 89 placed
< 90 placed
< 91 placed

Registers

> i figures+combining
> i combining
> m \tex {startcombina..
> m \tex {setupcombina..
> m \tex {placesidebys..
> m \tex {placeontopof..
```

12.3 Combining figures

For reasons of convenience we now discuss a command that enables us to combine floats into one.

```
\startcombination[...] ... \stopcombination
...     n*m
```

This command is used to place the figures under or next to each other.



92 **Figure 12.3** An example of `\startcombination...`.

The example in figure 12.3 is typeset with the commands:

```
\placefigure
  [here]
  [fig:combinations]
  {An example of \type{\startcombination...}.}
  {\startcombination[3*2]
     {\externalfigure[lb00220]} {a} {\externalfigure[lb00221]} {b}
     {\externalfigure[lb00222]} {c} {\externalfigure[lb00223]} {d}
     {\externalfigure[lb00225]} {e} {\externalfigure[lb00226]} {f}
   \stopcombination}
```

Between `[ ]` we specify how the combination is combined: `[3*2]`, `[4*2]` etc. When we put two floats next to each other it is sufficient to specify `[2]`, `[4]` etc.

The floats, mostly figures or tables, are specified within two arguments. The first content is placed over the second content: `{xxx}{yyy}`. The second argument can be empty: `{xxx}{}`. The general construct looks like this:

Floatblocks

< 92 placed

References

> stp:434
> stp:startcombination
> fig:combinations
< fig:combinations

Registers

> t \tex {startcombina..

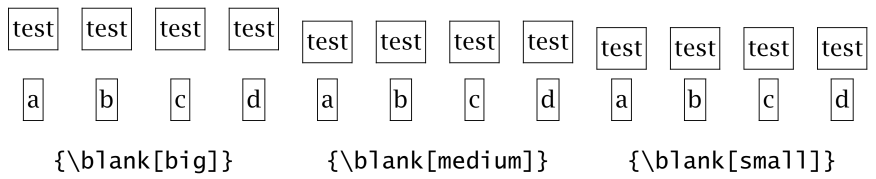
```
\startcombination[n*m]
  {text 1} {subcaption 1}
  {text 2} {subcaption 2}
  ........ ..............
\stopcombination
```

The combination can be set up with:

```
\setupcombinations[..,..=..,..]

before       commando
inbetween    commando
after        commando
distance     dimension
height       dimension fit
width        dimension fit
align        no left right middle normal
```

With `distance` you specify the horizontal distance between objects. The parameters `align` relates to the subcaption. By default the text and objects are centered. The width is the total width of the combination.

The three parameters `before`, `after` and `between` are processed in the order of specification in figure 12.5. There are some examples in figure 12.4. We can see in figure 12.6 that when the title in the second argument is empty the spacing adapted.



93 **Figure 12.4** The spacing within combinations (1).

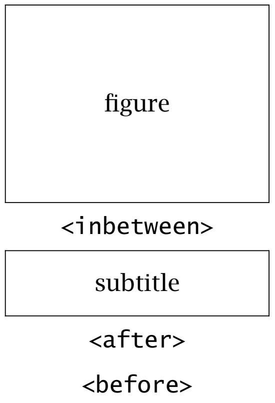
Using combinations require figures that have the correct dimensions or equal proportions. Unequally proportioned figures are hard to combine.

The simple version of combining is this:

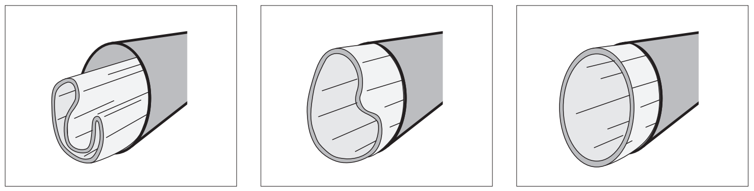
Floatblocks
< 93 placed
> 94 saved
> 95 saved

References
> stp:433
> stp:setupcombinations
< fig:order of combina..
< fig:spacing in combi..
< fig:no subcaptions
> fig:spacing in combi..
> fig:order of combina..
> fig:no subcaptions
> stp:435
> stp:placesidebyside

Registers
> t \tex {setupcombina..
> t \tex {placesidebys..



94 **Figure 12.5** The spacing within combinations (2).



95 **Figure 12.6** Combinations without captions.

```
\placesidebyside{.1.}{.2.}
.1.     text
.2.     text
```

```
\placeontopofeachother{.1.}{.2.}
.1.     text
.2.     text
```

We use them in this way:

```
\placesidebyside       {\framed{\Logo[ADE]}} {\framed{\Logo[BUR]}}
\placeontopofeachother {\framed{\Logo[ADE]}} {\framed{\Logo[BUR]}}
```

References

> stp:436
> stp:placeontopofeach..
> textblocks

Registers

> t \tex {placeontopof..
> i moving text
> i blocks+moving
> i blocks+numbering
> i numbering+blocks
> m \tex {defineblocks}
> m \tex {begin«block»}
> m \tex {hideblocks}
> m \tex {selectblocks}
> m \tex {useblocks}
> m \tex {keepblocks}
> m \tex {processblocks}
> m \tex {setupblock}
> m \tex {reset}

12.4 Text blocks

For practical reasons we sometimes want to key text somewhere in the source that should be typeset at a completely different location in the typeset document. It is also useful to be able to use text more than once. The commands described below are among the eldest of ConTEXt. They were one of the reasons to start writing the macropackage.

You can mark text (a text block) and hide or move that block, but first you have to define it using:

```
\defineblock[...]
...      name
```

If necessary you can pass several names in a comma–delimited list. After the definition you can mark text with:

```
\beginname
....................
....................
\endname
```

Between the `begin`– and `end` command you can use any command you want.

The commands below tell ConTEXt to hide or recall text blocks:

```
\hideblocks[..,.1.,..][..,.2.,..]
.1.      name
.2.      name
```

```
\useblocks[..,.1.,..][..,.2.,..]
.1.      name
.2.      name
```

```
\keepblocks[..,.1.,..][..,.2.,..]
.1.      name
.2.      all name
```

```
\selectblocks[..,.1.,..][..,.2.,..][..=..]
.1.         name
.2.         name
criterium   all section
```

```
\processblocks[..,.1.,..][..,.2.,..]
.1.     name
.2.     name
```

These commands make it necessary to process your text at least twice. You can also recall more than one text block, for example [question,answer].

In hidden and re-used blocks commands for numbering can be used. Assume that you use questions and answers in your document. By defining the questions as text blocks you can:

1. at that location typeset the questions
2. only use the questions and use the answers in a separate chapter
3. use questions and answers in a separate chapter
4. hide the answers
5. etc.

When we choose option 2 the definitions look like this:

```
\defineenumeration[question][location=top,text=Question]
\defineenumeration[answer][location=top,text=Answer]

\defineblock[question,answer]

\hideblocks[answer]
```

A question and answer in the source look like this:

```
\beginquestion
\question Why do we use blocks? \par
\endquestion

\beginanswer
\answer I really don’t know. \par
\endanswer
```

The questions are only used in the text. Questions and answers are both numbered. Answers are summoned by:

```
\chapter{Answers}

\reset[answer]
\useblocks[answer]
```

The command `\reset...` is necessary for resetting the numbering mechanism. When the answers are used in the same chapter you can use the following commands:

```
\section{Answers}

\reset[answer]
\selectblocks[answer][criterium=chapter]
```

You must be aware of the fact that it may be necessary to (temporarily) disable the reference mechanism also:

```
\setupreferencing[state=stop]
```

A more complex situation is this one. Assume that you have several mathematical formulas in your document, and that you want to recapitulate the more complex ones in a separate chapter at the end of the document. You have to specify an `[-]` at formulas you do not want repeated.

```
\defineblock[formula]

\beginformula
\placeformula[newton 1]$$f=ma$$
\endformula

This can also be written as:

\beginformula[-]
\placeformula[newton 2]$$m=f/a$$
\endformula
```

When you re–use the formulas only the first one is typeset. The rest of the formulas is processed, so the numbering will not falter.

The opposite is also possible. By default all local specifications are undone automatically. This means for example that the enumeration of text elements like questions, answers, definitions, etc. can be temporarily stopped. When numbering should continue you specify: `[+]`.[23]

Among the parameters of the number mechanism we (in some cases) use the parameter `blockwise`. This parameter relates to numbering within a set of blocks, for example per chapter.

[23] When you use enumerations within text blocks you can best use the `\start...stop` alternative (see page ??).

You may have a document in which the questions and answers are collected in text blocks. The questions are typeset in the document and the answers in a separate appendix. Answers and question are put at the same location in the source file. When we number the questions and answers per chapter, then question 4.12 is the 12th question in chapter 4. The correct number is used in the appendix. In this example answer 4.12 refers to question 4.12 and not the appendix number.

In case we do want the appendix number to be the prefix of the blocknumber we set the parameter `blockwise` at `no`. This is a rather complex situation and will seldom occur.

Earlier we discussed the initializing and resetting of counters. For reasons of uniformity we also have:

```
\reset[..,...,..]
...     name
```

In future there will be an option to sort blocks. For that purpose a second set of optional `[ ]` in and `\selectblocks` is available. The first argument is used for 'tags'. These tags are logical labels that enable us to recall the blocks.

```
\beginremark[important]
This is an important message!
\endremark
```

Now we can recall the 'important' messages by:

```
\useblocks[remark][important]
```

or:

```
\selectblocks[remark][important][criterium=chapter]
```

Here, `criterium` has the same function as in lists (like tables of content) and registers: it limits the search. In this case, only the blocks belonging to this chapter will be typeset.

More than one 'tag' is allowed in a comma delimited list. Text blocks may be nested:

```
\beginpractice
\beginquestion
\question Is that clear? \par
\endquestion
\beginanswer
```

```
\answer Yes it is! \par
\endanswer
\endpractice
```

In this case we use three blocks. Such blocks are stored in a file. This file must be available when the blocks are re-used. This means that the document must be processed at least twice. When blocks are summoned at the end of your source file only one processing step is sufficient but then you have to type the command `nomoreblocks` before the blocks are recalled:

```
\nomoreblocks
```

After this command no blocks should be specified. In the future commands will be developed for local adaptations of the layout of text blocks. Until that moment the following command is all there is:

```
\setupblock[..,...,..][..,..=..,..]

...       name
before    command
after     command
inner     command
style     normal bold slanted boldslanted type cap small... command
file      file
```

A block is being processed within a group, in other words: within `{ }`. The setup of `before` and `after` are used outside this group, and the setup of `inner` is used within the group. For example if we mark a re-used text block in the margin we can use the following setup:

```
\defineblock[exampletext]

\beginexampletext
If you wonder why this mechanism was implemented consider an educational
document with hundreds of \quote {nice to know} and \quote {need to know}
text blocks at several ability levels.
\endexampletext

\setupblock[exampletext][inner=\margintitle{reused}]
\useblocks[exampletext]
```

The first text is set without an indicator in the margin and the second is. If we would have used `before` instead of `inner` some grouping problems had occurred.

If you wonder why this mechanism was implemented consider an educational document with hundreds of ‘nice to know’ and ‘need to know’ text blocks at several ability levels.

reused If you wonder why this mechanism was implemented consider an educational document with hundreds of ‘nice to know’ and ‘need to know’ text blocks at several ability levels.

You can import text blocks from other source files. For example if you want to use text blocks from a manual for students in a manual for teachers, you can specify:

```
\setupblock
  [homework]
  [file=student,
   before=\startbackground,
   after=\stopbackground]
```

In that case the blocks are imported from the file `student.tex`. In this example these blocks are typeset differently, with a background. When the student material is specified with:

```
\beginhomework[meeting 1]
..........
\endhomework
```

we can summon the blocks in the teacher’s manual with:

```
\useblocks[homework][meeting 1]
```

In extensive documents it will take some time to generate these products. But this mechanism garantees we use the same homework descriptions in the students and teachers manual. Furthermore it saves typing and prevents errors.

Questions and answers are good examples of text blocks that can be hidden and moved. The example below will illustrate this. Because commands like `\question` have a paragraph as an argument the `\par`’s and/or empty lines are essential.

In the setup we see that questions and answers are coupled. A coupling has a meaning in interactive documents.

```
\defineblock[question]
\defineblock[answer]

\defineenumeration[question][location=inmargin,coupling=answer]
\defineenumeration[answer][location=top,coupling=question]

\hideblocks[answer]
```

```
\starttext

\chapter{\CONTEXT}

\CONTEXT\ is a macropackage that is based on \TEX. \TEX\ is a typesetting
system and a programm. This unique combination is used extensively in
\CONTEXT.

\beginquestion
  \startquestion
  To date, the fact that \TEX\ is a programming language enables \CONTEXT\
  to do text manipulations that cannot be done with any other known package.

  Can you mention one or two features of \CONTEXT\ that are based on the
  fact that \TEX\ is programming language?
  \stopquestion
\endquestion

\beginanswer
  \answer You can think of features like floating blocks and text block
  manipulation. \par
\endanswer

\beginquestion
  \question Are there any limitations in \TEX ? \par
\endquestion

\beginanswer
  \answer Yes and no. The implementation of \TEXEXEC\ is done in
  \PERL\ rather than in \TEX.
\endanswer

\TEX\ is a very powerful tool, but much of its power is yet to be
unleashed. \CONTEXT\ tries to make a contribution with its user||friendly
interface and its support of many features, like interactivety.

\chapter{Answers}

\useblocks[question,answer]

\stoptext
```

With `\processblocks` blocks are processed but not typeset. Assume that we have two types of questions:

```
\defineblock[easyquestion,hardquestion]
```

When both types of questions use the same numbering mechanism, we can recall the hard questions in their original order by hiding the easy questions.

```
\processblocks[easyquestion]
\useblocks[hardquestion]
```

12.5 Opposite blocks

In future versions of CONTEXT there will be support of spread based typesetting. For the moment the only command available is:

```
\startopposite ... \stopopposite
```

Everything between `start` and `stop` is typeset at the left page in such a way that it is aligned with the last paragraph that is typeset on the right page.

```
\setupoppositeplacing[..=..]
state    start stop
```

12.6 Margin blocks

Within limits you can place text and figures in the margin. In this case the margin is handled as a separate (very narrow) page next to the actual page.

```
\startmarginblock ... \stopmarginblock
```

This can be setup with:

```
\setupmarginblocks[..,..=..,..]

location     inmargin left middle right
style        normal bold slanted boldslanted type cap small... command
width        dimension
align        left middle right no
top          command
inbetween    command
bottom       command
left         command
right        command
before       command
after        command
```

The mechanism to place blocks is still under construction.

12.7 Hiding text

It is possible to hide text (skip during processing) by:

```
\starthiding ... \stophiding
```

12.8 Postponing text

Text elements can be postponed (stored) and placed at the next empty page. This option is needed in case CONTEXT encounters large figures or tables. The postponed textelement is placed at the next page generated by TEX or forced by the user with a manual page break.

```
\startpostponing ... \stoppostponing
```

Several text blocks can be postponed and stored. This proces can be followed on screen during document generation.

```
\startpostponing
\placefigure{A rather large figure.}{...}
\stoppostponing
```

When a lot of text elements are postponed or when a figure uses a complete page we advise you to add `\page` after the postponing. Otherwise there is the possibility that a blank page is inserted. This is caused by the fact that the postponing mechanism and the float mechanism are completely independent.

```
\startpostponing
\placefigure{A very large figure.}{...}
\page
\stoppostponing
```

12.9 Buffers

Buffers simplify the moving of text blocks. They are stored in a file with the extension `tmp` and are used to bring readability to your source. Furthermore they can be recalled at any location without retyping them.

```
\startbuffer[...] ... \stopbuffer
...     name
```

```
\getbuffer[...]
...     name
```

```
\typebuffer[...]
...     name
```

The example below shows the use of these commands.

```
\startbuffer
We see that a {\em buffer} works something like a {\em block}.\par
\stopbuffer
```

```
\startlines
{\tf \getbuffer}
{\bf \getbuffer}
{\sl \getbuffer}
\stoplines
```

This results in:

We see that a *buffer* works something like a *block*.
We see that a *buffer* works something like a *block*.
We see that a buffer *works something like a* block.

The name is optional. A name makes sense only when several buffers are used. Most of the time the default buffer will do. Most examples in this manual are typed in buffers.

In chapter ?? we can see that the last argument of a `\place`*`block`* can be rather extensive. A buffer can be useful when such large tables are defined.

```
\startbuffer
...  many lines ...
\stopbuffer
```

```
\placetable{A table.}{\getbuffer}
```

The buffer is set up with:

```
\setupbuffer[...][..,..=..,..]

...       name
before    command
after     command
```

The first argument is optional and relates to the buffers you defined yourself. You can define your own buffer with:

```
\definebuffer[...]

...      name
```

Be aware of possible conflicting names and use capital letters. After this command `/get`*`buffer`* and `/type`*`buffer`* are available where *`buffer`* is the name of the buffer.

\externalfigure	*299*, *304*	\showexternalfigures	*304*
\setupexternalfigures	*299*	\useexternalfigure	*299*

12 Buffers

Figures

13

13.1 Introduction

In this chapter we discuss how to place figures in your document. In section 12.2 we introduced the float mechanism. In this chapter the placement of figures is discussed. Most of the time these figures are created with external applications.

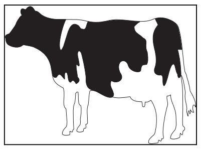
After processing a document the result is a DVI file or, when we use PDFTEX, a PDF file. The DVI document reserves space for the figure, but the figure itself will be put in the document during postprocessing of the DVI file. PDFTEX needs no postprocessing and the external figures are automatically included in the PDF file.

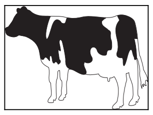
External figures may have different formats like the vector formats EPS and PDF, or the bitmap formats TIF, PNG and JPG. Note that we refer to figures but we could also refer to movies. ConTEXt has special mechsnisms to handle figures generated by METAPOST. We have to take care that fonts used in METAPOST figures are recognized by PDFTEX. Finally, we'll see that METAPOST code can be embedded in ConTEXt documents.

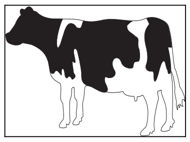
Normally, users need not concern themselves with the internal mechanisms used by ConTEXt for figure processing. However some insight may be useful.

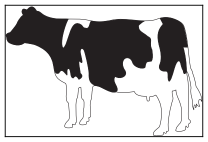
13.2 Defining figures

A figure is designed within specific dimensions. These dimensions may of may not be known by the document designer.


natural
dimension


scaled
to 25%


a height
of 2 cm


a height of 2 cm
and a width 3 cm

If the original dimensions are unknown, then scaling the figure to 40% can have some astonishing results. A figure with width and height of 1 cm becomes almost invisible, but a figure width width and height of 50 cm will still be very large when scaled to 40% of its original size.

A better strategy is to perform the scaling based on the current bodyfont size, the width of text on the page, or to set absolute dimensions, such as 3 cm by 2 cm.

To give TEX the opportunity to scale the figure adequately the file format must be known. Table 13.1 shows the file formats supported by DVIPS, DVIPSONE, and PDFTEX respectively. PDFTEX has the unique capability to determine the file format during processing.

When we use DVI, TEX can determine the dimensions of an EPS illustration by searching for the so called *bounding box*. However, with other formats such as TIF, the user is responsible for the determination of the figure dimensions.

	EPS	PDF	METAPOST	TIF	PNG	JPG	MOV
DVIPS	+	–	+	-	-	-	+
DVIPSONE	+	–	+	+	-	-	+
PDFTEX	-	+	+	+	+	+	+

Table 13.1 Some examples of supported file formats.

Now, let us assume that the dimensions of a figure are found. When we want to place the same figure many times, it would be obvious to search for these dimensions only once. That is exactly what happens. When a figure is found it is stored as an object. Such an object is re–used in TEX and in PDF but not in DVI, since reuse of information is not supported by the DVI format. To compensate for this shortcoming, when producing DVI output, CONTEXT will internally reuse figures, and put duplicates in the DVI file.

```
\useexternalfigure[some logo][logo][width=3cm]

\placeexternalfigure{first logo}{\externalfigure[some logo]}

\placeexternalfigure{second logo}{\externalfigure[some logo]}
```

So, when the second logo is placed, the information collected while placing the first one is used. In PDFTEX even the content is reused, if requested, at a different scale.

A number of characteristics of external figures are specified by:

```
\setupexternalfigures[..=..]

option      frame empty test
object      yes no
frames      on off
ymax        number
xmax        number
directory   text
location    local global
```

This command affect all figures that follow. Three options are available: `frame`, `empty` and `test`. With `empty` no figures are placed, but the necessary space is reserved. This can save you some time when 'testing' a document.[24] Furthermore the figure characteristics are printed in that space. When `frame` is set at `on` a frame is generated around the figure. The option `test` relates to testing hyperactive areas in figures.

When CONTEXT is not able to determine the dimensions of an external figure directly, it will fall back on a simple database that can be generated by the PERL script TEXUTIL. You can generate such a database by calling this script as follows:

```
texutil  --figures  *.tif
```

This will generate the `texutil.tuf` file, which contains the dimensions of the TIF figures found. You need to repeat this procedure every time you change a graphic. Therefore, it can be more convenient to let CONTEXT communicate with TEXUTIL directly. You can enable that by adding `\runutilityfiletrue` to your local `cont-sys.tex` file.

When a figure itself is not available but it is listed in the `texutil.tuf` file then CONTEXT presumes that the figure does exist. This means that the graphics do not need to be physically present on the system.

Although CONTEXT very hard tries to locate a figure, it may fail due to missing or invalid figure, or invalid path specifications (more on that later). The actual search depends on the setup of directories and the formats supported. In most cases, it it best not to specify a suffix or type.

```
\exernalfigure[hownice]
\exernalfigure[hownice.pdf]
\exernalfigure[hownice][type=pdf]
```

[24] A similar effect can be obtained with the `--fast` switch in TEXEXEC.

In the first case, ConTeXt will use the graphic that has the highest quality, while in both other cases, a PDF graphic will be used. In most cases, the next four calls are equivalent, given that `hownice` is available in METAPOST output format with a suffix `eps` or `mps`:

```
\exernalfigure[hownice]
\exernalfigure[hownice][type=eps]
\exernalfigure[hownice][type=eps,method=mps]
\exernalfigure[hownice][type=mps]
```

In most cases, a METAPOST graphic will have a number as suffix, so the next call makes the most sense:

```
\exernalfigure[hownice.1]
```

Let us summarize the process. Depending on the formats supported by the currently selected driver (DVI, PDFTeX, etc.), ConTeXt tries to locate the graphics file, starting with the best quality. When found, ConTeXt first tries to determine the dimensions itself. If this is impossible, ConTeXt will look into `texutil.tuf`. The graphic as well as the file `texutil.tuf` are searched on the current directory (`local`) and/or dedicated graphics directories (`global`), as defined by `\setupexternalfugures`. By default the `location` is set at `{local,global}`, so both the local and global directories are searched. You can set up several directories for your search by providing a comma–delimited list:

```
\setupexternalfigures[directory={c:/fig/eps,c:/fig/pdf}]
```

Even if your operating uses a \ as separator, you should use a `/`. The figure directory may be system dependent and is either set in the file `cont-sys`, in the document preamble, or in a style.

An external figure is summoned by the command `\externalfigure`. The cow is recalled with:

```
\exernalfigure[koe][width=2cm]
```

For reasons of maintenance it is better to specify all figures at the top of your source file or in a separate file. The figure definition is done with:

`\useexternalfigure`*`[.1.]`*`[.2.]`*`[..,..=..,..]`*

.1.	*name*
.2.	*file*
scale	*number*
factor	max fit broad
wfactor	*number* max broad fit
hfactor	*number* max broad fit
width	*dimension*
height	*dimension*
frame	on off
preset	yes no
display	*file*
preview	yes no
repeat	yes no
object	yes no
type	eps mps pdf tif png jpg mov tex
method	eps mps pdf tif png jpg mov tex

Valid definitions are:

```
\useexternalfigure [cow]
\useexternalfigure [some cow] [cow230]
\useexternalfigure [big cow]  [cow230] [width=4cm]
```

In the first definition, the figure can be recalled as `cow` and the graphics file is also `cow`. In the second and third definition, the symbolic name is `some cow`, while the filename is `cow230`. The last example also specifies the dimensions.

The `scale` is given in percentages. A scale of `800` (80%) reduces the figure, while a value of `1200` (120%) enlarges the figure. Instead of using percentages you can also scale with a factor that is related to the actual bodyfont. A setup of `hfactor=20` supplies a figure with 2 times the height of the bodyfont size, and `bfactor=120` will result in a width of 12 times the bodyfont size (so 144pt when using a 12pt bodyfont size). When we want to place two figures next to one another we can set the height of both figures with `hfactor` at the same value:

```
\useexternalfigure[alfa][file0001][hfactor=50]
\useexternalfigure[beta][file0002][hfactor=50]
```

```
\placefigure
  {Two figures close to one another.}
  \startcombination[2]
    {\externalfigure[alfa]} {this is alfa}
    {\externalfigure[beta]} {this is beta}
  \stopcombination
```

We can see that `\externalfigure` is capable of using a predefined figure. The typographical consistency of a figure may be enhanced by consistently scaling the figures. Also, figures can inherit characteristics of previously defined figures:

```
\useexternalfigure [alfa]  [file0001] [hfactor=50]
\useexternalfigure [beta]  [file0002] [alfa]
\useexternalfigure [gamma] [file0003] [alfa]
\useexternalfigure [delta] [file0004] [alfa]
```

Normalizing a figure's width must also be advised when figures are placed with `\startfiguretext` below one another.

In most cases you will encounter isolated figures of which you want to specify width or height. In that case there is no relation with the bodyfont except when the units `em` or `ex` are used.

In figure 13.1 we drew a pattern with squares of a factor 10.

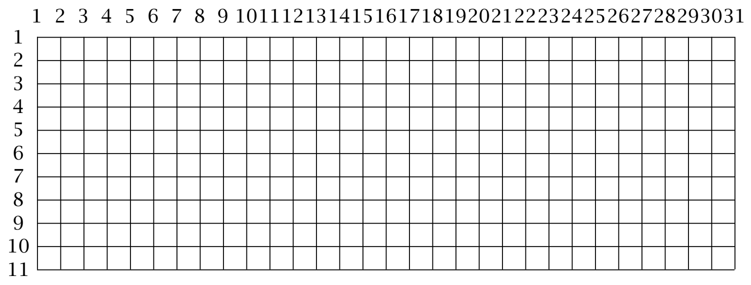


Figure 13.1 Factors at the actual bodyfont.

13.3 Recalling figures

A figure is recalled with the command:

```
\externalfigure[...][..,..=..,..]

...      file
..=..    see p 303: \useexternalfigure
```

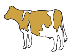
For reasons of downward compatibility a figure can also be recalled with a command that equals the figure name. In the example below we also could have used `\akoe` and `\bkoe`, unless they are already defined. Using `\externalfigure` instead is more safe, since it has its own namespace.

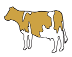
```
\useexternalfigure[akoe][koetje][factor=10]
\useexternalfigure[bkoe][koetje][factor=20]
```

```
\placefigure[left]{none}{\externalfigure[bkoe]}
```

```
The \hbox {\externalfigure[akoe]} is a very well known animal in the Dutch
landscape. But for environmental reasons the \hbox {\externalfigure[akoe]}
is slowly disappearing. In the near future the cow will fulfil a marginal
\inleft {\externalfigure[bkoe]} role in the Netherlands. That is the
reason why we would like to write the word \hbox {\externalfigure[bkoe]}
in big print.
```

Here we see how `akoe` and `bkoe` are reused. This code will result in:



The is a very well known animal in the Dutch landscape. But for environmental reasons the is slowly disappearing. In the near future the cow will fulfil a marginal  role in the Netherlands. That is the reason why we would like to write the word in big print.

Normalized figures adapt to the actual bodyfont at least when the font is set with `\setupbodyfont` or `\switchtobodyfont`. When a text is used for different media and is generated with different fontsizes the use of normalized figures is a good practice. The example above looks different in a smaller fontsize.

The is a very well known animal in the Dutch landscape. But for environmental reasons the is slowly disappearing. In the near future the cow will fulfil a marginal role in the Netherlands. That is the reason why we would like to write the word in big print.

13.4 Automatic scaling

In cases where you want the figure displayed as big as possible you can set the parameter `factor` at `max`, `fit` or `broad`. In most situations the value `broad` will suffice, because then the caption still fits on a page.

setup	result
`max`	maximum width or height
`fit`	remaining width or height
`broad`	more remaining width or height
`number`	scaling factor (times 10)

Table 13.2 Normalized figures.

So, one can use `max` to scale a figure to the full page, or `fit` to let it take up all the remaining space. With `broad` some space is reserved for a caption.

Sometimes it is not clear whether the height or the width of a figure determines the optimal display. In that case you can set `factor` at `max`, so that the maximal dimensions are determined automatically.

```
\externalfigure[cow][factor=max]
```

This figure of a cow will scale to the width or height of the text, whichever fits best. Even combinations of settings are possible:

```
\externalfigure[cow][factor=max,height=.4\textheight]
```

In this case, the cow will scale to either the width o fthe text or 40% of the height of the text, depending on what fits best.

As already said, the figures and their characteristics are stored in the file `texutil.tuf` and can be displayed with:

```
\showexternalfigures[..,..=..,..]
alternative    a b c
```

There are two alternatives: `a`, `b` and `c`. The first alternative leaves room for figure corrections and annotations, the second alternative is somewhat more efficient and places more figures on one page. The third alternative puts each figure on its own page. Of course one needs to provide the file `texutil.tuf` by saying:

```
texutil --figures *.mps *.jpg *.png
```

Even more straightforward is running TEXEXEC, for instance:

```
texexec --figures=c --pdf *.mps *.jpg *.png
```

This will give you a PDF file of the figures requested, with one figure per page.

13.5 TEX-figures

Figures can be scaled. This mechanism can also be used for other text elements. These elements are then stored in separate files or in a buffer. The next example shows how a table is scaled to the pagewidth. The result is typeset in figure 13.2.

```
\startbuffer[table]
  \starttable[||||||]
    \HL
    \VL \bf factor           \VL \bf width            \VL
        \bf height           \VL \bf width and height \VL
        \bf nothing          \VL \SR
    \HL
    \VL \type{max}           \VL automatically        \VL
        automatically        \VL automatically        \VL
        width or height      \VL \FR
    \VL \type{fit}           \VL automatically        \VL
        automatically        \VL automatically        \VL
        width or height      \VL \MR
    \VL \type{broad}         \VL automatically        \VL
        automatically        \VL automatically        \VL
        width or height      \VL \MR
    \VL \type{...}           \VL width                \VL
        height               \VL isometric            \VL
        original dimensions  \VL \LR
    \HL
  \stoptable
\stopbuffer
```

```
\placefigure
  [here][fig:table]
  {An example of a \TEX\ figure.}
  {\externalfigure[table.tmp][width=\textwidth]}
```

```
\placefigure
  {An example of a \TEX\ figure.}
```

```
{\externalfigure[table.tmp][width=.5\textwidth]}
```

factor	**width**	**height**	**width and height**	**nothing**
`max`	automatically	automatically	automatically	width or height
`fit`	automatically	automatically	automatically	width or height
`broad`	automatically	automatically	automatically	width or height
`...`	width	height	isometric	original dimensions

Figure 13.2 An example of a TEX figure.

factor	**width**	**height**	**width and height**	**nothing**
`max`	automatically	automatically	automatically	width or height
`fit`	automatically	automatically	automatically	width or height
`broad`	automatically	automatically	automatically	width or height
`...`	width	height	isometric	original dimensions

Figure 13.3 An example of a TEX figure.

Buffers are written to a file with the extension `tmp`, so we recall the table with `table.tmp`. Other types of figures are searched on the directories automatically. With TEX figures this is not the case. This might lead to conflicting situations when an EPS figure is meant and not found, but a TEX file of that name is.

13.6 Extensions of figures

In the introduction we mentioned different figure formats like EPS and PNG. In most situations the format does not have to be specified. On the contrary, format specification would mean that we would have to re-specify when we switch from DVI to PDF output. The figure format that CONTEXT will use depends on the special driver. First preference is an outline, second a bitmap.

METAPOST figures, that can have a number as suffix, are recognized automatically. CONTEXT will take care of the font management when it encounters METAPOST figures. When color is disabled, or RGB is to be converted to CMYK, CONTEXT will determine what color specifications have to be converted in the METAPOST file. If needed, colors are converted to weighted grey scales, that print acceptable on black and white printers. In the next step the fonts are smuggled into the file.[25] In case of PDF output the METAPOST code is converted into PDF by TEX.

If necessary the code needed to insert the graphic is stored as a so called object for future re-use. This saves processing time, as well as bytes when producing PDF. You can prevent this by setting `object=no`.

When EPS and MPS (METAPOST) figures are processed CONTEXT searches for the high resolution bounding box. By default the POSTSCRIPT bounding box may have a deviation of half a point, which is within the accuracy of our eyes. Especially when aligning graphics, such deviations will not go unnoticed.

CONTEXT determines the file format automatically, as is the case when you use:

```
\externalfigure[koe]
```

Sometimes however, as we already explained, the user may want to force the format for some reason. This can be done by:

```
\externalfigure[koe.eps]
\externalfigure[koe][type=eps]
```

In special cases you can specify in which way figure processing takes place. In the next example CONTEXT determines dimensions asif the file were in EPS format, that is, it has a bounding box, but processes the files as if it were a METAPOST file. This kind of detailed specification is seldom needed.

```
\externalfigure[graphic.xyz][type=eps,method=mps]
```

The automatic searching for dimensions can be blocked by `preset=no`.

13.7 Movies

In CONTEXT moving images or 'movies' are handled just like figures. The file format type is not determined automatically yet. This means the user has to specify the file format.

```
\externalfigure[demo.mov][label=demo,width=4cm,height=4cm,preview=yes]
```

With this setup a preview is shown (the first image of the movie). If necessary an ordinary (static) figure can be layed over the first movie image with the overlay mechanism.

[25] Fonts are a problem in METAPOST files, since it it up to the postprocessor to take care of them. In this respect, METAPOST output is not self contained.

Movies can be controlled either by clicking on them, or by providing navigational tools, like:

```
... \goto {start me} [StartMovie{demo}] ...
```

A more detailed discussion on controlling widgets is beyond this chapter. Keep in mind that you need to distribute the movies along with your document, since they are not included. This makes sense, since movies can be pretty large.

13.8 Some remarks on figures

Figures, and photos in particular, have to be produced with consistent proportions. The proportions specified in figure 13.4 can be used as a guideline. Scaling of photos may cause quality loss.

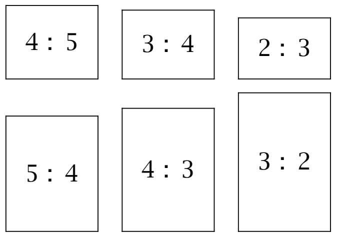


Figure 13.4 Some preferred image proportions.

In the background of a figure you typeset a background (see figure ??). In this example the external figures get a background (for a black and white reader: a green screen).

```
\setupfloats
  [background=color,
   backgroundcolor=green,
   backgroundoffset=3pt]
\useexternalfigure [koe]
  [bfactor=80,
   background=screen,
   backgroundscreen=0.75]
```

Note that we use only one float and that there are six external figures. The background of the float is used for the complete combination and the background of the external figure only for the figure itself.

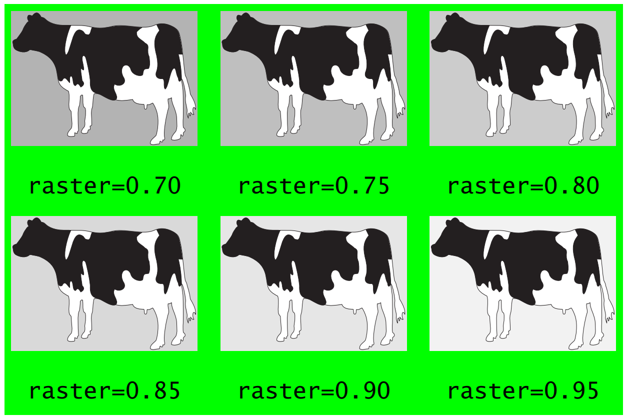


Figure 13.5 Some examples of backgrounds in figures.

Definitions

```
\about{...}[ref]                                              ◄ ►  209
...      text
```

```
\adaptlayout[..,...,..][..,..=..,..]                                34
...       number
height    dimension max
lines     number
```

```
\arg{...}                                                          126
...      text
```

```
\at{.1.}{.2.}[ref]                                            ◄ ►  209
.1.      text
```

```
\atpage[ref]                                                       210
```

```
\background                                                        152
```

```
\blackrule[..,..=..,..]                                            271
..=..    see p 272: \setupblackrules
```

```
\blackrules[..,..=..,..]                                           272
..=..    see p 272: \setupblackrules
```

```
\blank[..,...,..]                                                   69
...      n*small n*medium n*big nowhite back white disable force reset line halfline formula fixed flexible
```

```
\but[ref]                                                     ◄ ►  243
```

```
\Cap{...}                                                          119
...      text
```

\CAP{...} **120**

... *text*

\Caps{.. ... ..} **120**

... *text*

\chapter*[ref,..]*{...} **171**

... *text*

\color[.1.]{.2.} **143**

.1. *text*

\column **79**

\comparecolorgroup[...] **149**

... *name*

\comparepalet[...] **149**

... *name*

\complete*combinedlist*[...][..,..=..,..] **190**

... *name*
..=.. see p 190: \setupcombinedlist

\completelistof*floats* ◄ ► **279**

\completelistof*sorts* **203**

\completelistof*synonyms* **202**

\complete*register[..,..=..,..]* ◄ ► **217**

..=.. see p 218: \setupregister

\correctwhitespace{...} **72**

A

```
\coupledregister[.1.]{.2.}                                222
.1.     text
.2.     text
```

```
\couplemarking[.1.][.2.]                                  208
.1.     name
.2.     name
```

```
\coupleregister[...]                                      222
...     name
```

```
\crlf                                                      89
```

```
\currentdate[..,...,..]                                   163
...     day month MONTH weekday WEEKDAY year dd mm jj yy d m j y referral
```

```
\currentheadnumber                                        176
```

```
\date[..,..=..,..][..,...,..]                             163
d       number
m       number
y       number
...     see p 163: \currentdate
```

```
\decouplemarking[...]                                     208
...     name
```

```
\defineblank[.1.][.2.]                                     71
.1.     name
.2.     see p 70: \setupblank
```

```
\defineblock[...]                                         287
...     name
```

\definebodyfont[..,.1.,..][.2.][..,..=..,..] **133**

.1.	5pt ... 12pt default
.2.	rm ss tt mm hw cg
tf	*file*
bf	*file*
sl	*file*
it	*file*
bs	*file*
bi	*file*
sc	*file*
ex	*file*
mi	*file*
sy	*file*
ma	*file*
mb	*file*
mc	*file*

\definebuffer[...] **297**

...	*name*

\definecolor[...][..,..=..,..] **143**

...	*name*
r	*text*
g	*text*
b	*text*
c	*text*
m	*text*
y	*text*
k	*text*
s	*text*

\definecolorgroup[.1.]*[.2.]*[..,.3.,..] **146**

.1.	*name*
.2.	rgb cmyk gray s
.3.	[x:y:z=,..]

\definecombinedlist[.1.][..,.2.,..]*[..,..=..,..]* **190**

.1.	*name*
.2.	*list*
..=..	see p 190: \setupcombinedlist

\definedescription[...]*[..,..=..,..]* **225**

...	*name*
..=..	see p 227: \setupdescriptions

A

\defineenumeration[..,.1.,..]*[.2.][..,..=..,..]* **228**

.1.	*name*
.2.	*name*
..=..	see p 230: \setupenumerations

\definefloat[.1.][.2.] **276**

.1.	*singular name*
.2.	*plural name*

\defineframedtext[...]*[..,..=..,..]* **269**

...	*name*
..=..	see p 267: \setupframedtexts

\definehead[.1.][.2.] **173**

.1.	*name*
.2.	*section*

\defineindenting[...][..,..=..,..] **232**

...	*name*
..=..	see p 233: \setupindentations

\definelabel[...][..,..=..,..] **234**

...	*name*
text	*text*
location	inmargin intext
way	bytext by*section* bychapter
blockway	yes no
headstyle	normal bold slanted boldslanted type cap small... *command*
headcolor	*name*
before	*command*
after	*command*

\definelist[.1.]*[.2.][..,..=..,..]* **187**

.1.	*name*
.2.	*name*
..=..	see p 189: \setuplist

\definelogo[.1.][.2.][.3.][..,..=..,..] **57**

.1.	*name*
.2.	top header footer bottom
.3.	none page leftedge leftmargin left middle right rightmargin rightedge
command	*command text*
state	start stop

A

`\definemakeup[...][..,..=..,..]` **107**

...	*name*
..=..	see p 108: \setupmakeup

`\definemarking[.1.]`*`[.2.]`* **205**

.1.	*name*
.2.	*name*

`\definepalet[...][..,..=..,..]` **147**

...	*name*
name	*name*

`\definepapersize[...][..,..=..,..]` **30**

...	*name*
width	*dimension*
height	*dimension*
offset	*dimension*
scale	*number*

`\defineparagraphs[...][..,..=..,..]` **82**

...	*name*
n	*number*
rule	on off
height	fit *dimension*
before	*command*
after	*command*
inner	*command*
distance	*dimension*
tolerance	verystrict strict tolerant verytolerant stretch
align	left right middle

`\definereference[...][`*`ref`*`,..]` **214**

...	*name*

`\definereferenceformat[...][..,..=..,..]` **215**

...	*name*
left	*text*
right	*text*
text	*text*
label	*name*

`\defineregister[.1.][.2.]` **216**

.1.	*singular name*
.2.	*plural name*

```
\definesection[...]                                              182
...      name
```

```
\definesectionblock[...][..,..=..,..]                            182
...      name
..=..    see p 182: \setupsectionblock
```

```
\definesorting[.1.][.2.][.3.]                                    203
.1.      singular name
.2.      plural name
.3.      command
```

```
\definesynonyms[.1.][.2.][.3.][.4.]                              200
.1.      singular name
.2.      plural name
.3.      command
.4.      command
```

```
\definetext[.1.][.2.][.3.][.4.][.5.]                             100
.1.      name
.2.      header footer
.3.      text
.4.      text
.5.      text
```

```
\definetyping[...][..,..=..,..]                                  125
...      file typing name
..=..    see p 123: \setuptyping
```

```
\determineheadnumber[...]                                        176
...      section
```

```
\determinelistcharacteristics[..,...,..][..,..=..,..]            194
...      name
..=..    see p 189: \setuplist
```

```
\externalfigure[...][..,..=..,..]                                304
...      file
..=..    see p 303: \useexternalfigure
```

```
\fillinline[..,..=..,..]...\par                                   253
..=..    see p 254: \setupfillinlines
```

```
\fillinrules[..,..=..,..]{.1.}{.2.}                               254
..=..    see p 254: \setupfillinrules
```

```
\fixedspaces                                                      73
```

```
\footnote[ref]{...}                                               100
...      text
```

```
\framed[..,..=..,..]{...}                                         258
..=..    see p 264: \setupframed
...      text
```

```
\getbuffer[...]                                                   296
...      name
```

```
\getmarking[.1.][.2.]                                             205
.1.      name
.2.      first last previous both all
```

```
\godown[...]                                                      72
...      dimension
```

```
\graycolor[...]                                                   145
...      text
```

A

`\grid[..,..=..,..]` **273**

x	*number*
y	*number*
nx	*number*
ny	*number*
dx	*number*
dy	*number*
xstep	*number*
ystep	*number*
offset	yes no
factor	*number*
scale	*number*
unit	cm pt em mm ex sp in
location	left middle

`\hairline` **251**

`\head`*`[ref,..]`* **241**

`\headnumber`*`[...]`* **176**

...	*section*

`\headtext{...}` **165**

...	*text*

`\hideblocks[..,.1.,..]`*`[..,.2.,..]`* **287**

.1.	*name*
.2.	*name*

`\high{...}` **77**

...	*text*

`\hl[...]` **253**

...	*number*

`\in{.1.}{.2.}[`*`ref`*`]` ◀ ▶ **208**

.1.	*text*

`\indenting[..,...,..]` **67**

...	never not no yes always first next

`\inleft[.1.][ref]{.2.}` **74**

.1.	+ – low
.2.	*text*

`\inline[ref]` **214**

`\inmargin[.1.][ref]{.2.}` **74**

.1.	+ – low
.2.	*text*

`\inothermargin[.1.][ref]{.2.}` **74**

.1.	+ – low
.2.	*text*

`\inright[.1.][ref]{.2.}` **74**

.1.	+ – low
.2.	*text*

`\installlanguage[...][..,..=..,..]` **162**

...	*name*
spacing	packed broad
lefthyphenmin	*dimension*
righthyphenmin	*dimension*
state	start stop
leftsentence	*command*
rightsentence	*command*
leftsubsentence	*command*
rightsubsentence	*command*
leftquote	*command*
rightquote	*command*
leftquotation	*command*
rightquotation	*command*
default	*name*

`\item[ref,..]` **242**

`\items[..,..=..,..]{..,...,..}` **246**

..=..	see p 245: \setupitems

`\its[ref,..]` **242**

A

```
\kap{...}                                                  119
...      text
```

```
\keepblocks[..,.1.,..][..,.2.,..]                          287
.1.      name
.2.      all name
```

```
\labeltext{...}                                            164
...      text
```

```
\language[...]                                             159
...      nl fr en uk de sp cz ..
```

```
\leftaligned{...}                                           87
...      text
```

```
\loadsorts                                                 204
```

```
\loadsynonyms                                              202
```

```
\lohi{.1.}{.2.}                                             78
.1.      text
.2.      text
```

```
\low{...}                                                   77
...      text
```

```
\mainlanguage[...]                                         165
...      nl fr en uk de sp cz ..
```

```
\mar[ref,..]{...}                                          242
```

```
\marginrule[.1.]{.2.}                                      270
.1.      number
```

```
\margintext[.1.][ref]{.2.}                                76
.1.      + - low
.2.      text
```

```
\marking[.1.]{.2.}                                       205
.1.      name
.2.      text
```

```
\midaligned{...}                                          87
...      text
```

```
\moveongrid[...]                                          44
...      top both bottom
```

```
\nocap{...}                                              120
...      text
```

```
\noheaderandfooterlines                                   98
```

```
\noindenting                                              67
```

```
\nolist{...}                                             192
...      text
```

```
\nomarking{...}                                          173
...      text
```

```
\nomoreblocks                                            291
```

```
\nop                                                     244
```

```
\nospace                                                  73
```

```
\note[ref]                                               100
```

```
\notopandbottomlines                                      99
```

```
\nowhitespace                                                        69
```

```
\overbar{...}                                                       257
...      text
```

```
\overbars{..  ... ..}                                               257
...      text
```

```
\overstrike{...}                                                    258
...      text
```

```
\overstrikes{..  ... ..}                                            258
...      text
```

```
\packed                                                              71
```

```
\page[..,...,..]                                                     91
...      yes markup no preference bigpreference left right disable last quadruple even odd blank empty reset
```

```
\pagereference[ref]                                                 209
```

```
\part[ref,..]{...}                                                  171
...      text
```

```
\placecombinedlist[...][..,..=..,..]                                190
...      name
..=..    see p 190: \setupcombinedlist
```

```
\placefloat[.1.][ref,..]{.2.}{.3.}                                  276
.1.      left right here top bottom inleft inright inmargin margin page opposite always force tall
.2.      text
.3.      text
```

```
\placefootnotes[..,..=..,..]                                        103
..=..    see p 102: \setupfootnotes
```

A

\placelist[..,...,..]*[..,..=..,..]* **188**

...	*name*
..=..	see p 189: \setuplist

\placelistof*floats* ◀ ▶ **279**

\placelistof*sorts* **203**

\placelistof*synonyms* **202**

\placelocalfootnotes*[..,..=..,..]* **103**

..=..	see p 102: \setupfootnotes

\placelogos*[..,...,..]* **57**

...	*name*

\placeongrid*[.1.]*{.2.} **44**

.1.	see p 44: \moveongrid

\placeontopofeachother{.1.}{.2.} **286**

.1.	*text*
.2.	*text*

\place*register[..,..=..,..]* ◀ ▶ **217**

..=..	see p 218: \setupregister

\placesidebyside{.1.}{.2.} **286**

.1.	*text*
.2.	*text*

\processblocks[..,.1.,..]*[..,.2.,..]* **288**

.1.	*name*
.2.	*name*

\quotation{...} **247**

...	*text*

A

`\quote{...}` **247**

...	*text*

`\ran{...}` **242**

`\reference[ref]{...}` **209**

...	*text*

`\reservefloat[..,..=..,..][.1.][ref,..]{.2.}` **278**

height	*dimension*
width	*dimension*
frame	on off
.1.	left right here top bottom inleft inright inmargin margin page opposite always force
.2.	*text*

`\reset[..,...,..]` **290**

...	*name*

`\resetmarking[...]` **206**

...	*name*

`\rightaligned{...}` **87**

...	*text*

`\section[ref,..]{...}` **171**

...	*text*

`\seeregister[.1.]{.2.}{..+.3.+..}` **217**

.1.	*text*
.2.	*text*
.3.	*text*

`\selectblocks[..,.1.,..][..,.2.,..][..=..]` **288**

.1.	*name*
.2.	*name*
criterium	all *section*

`\setupalign[...]` **87**

...	width left right middle inner outer wide broad height bottom line reset

`\setuparranging[..,...,..]` **48**

...	disable 2*16 2*8 2*4 2*2 2UP 2DOWN mirrored rotated doublesided negative 90 180 270

`\setupbackground[..,..=..,..]` **151**

leftoffset	*dimension*
rightoffset	*dimension*
topoffset	*dimension*
bottomoffset	*dimension*
state	start stop
..=..	see p 264: \setupframed

`\setupbackgrounds`*`[.1.][..,.2.,..]`*`[..,..=..,..]` **152**

.1.	top header text footer bottom page paper leftpage rightpage
.2.	leftedge leftmargin text rightmargin rightedge
state	start stop repeat
..=..	see p 264: \setupframed

`\setupblackrules[..,..=..,..]` **272**

width	*dimension* max
height	*dimension* max
depth	*dimension* max
alternative	a b
distance	*dimension*
n	*number*

`\setupblank`*`[...]`* **70**

...	normal standard line *dimension* big medium small fixed flexible

`\setupblock[..,...,..][..,..=..,..]` **291**

...	*name*
before	*command*
after	*command*
inner	*command*
style	normal bold slanted boldslanted type cap small... *command*
file	*file*

`\setupbodyfont[..,...,..]` **114**

...	*name* serif regular roman sans support sansserif mono type teletype handwritten calligraphic 5pt ... 12pt

`\setupbodyfontenvironment[...][..,..=..,..]` **130**

...	see p 114: \setupbodyfont
..=..	see p 114: \setupbodyfont

\setupbottom*[...]*[..,..=..,..] **99**

...	see p 97: \setupheader
..=..	see p 97: \setupheader

\setupbottomtexts*[.1.]*[.2.][.3.] **99**

.1.	text margin edge
.2.	*text* *section* date *mark* pagenumber
.3.	*text* *section* date *mark* pagenumber

\setupbuffer*[...]*[..,..=..,..] **297**

...	*name*
before	*command*
after	*command*

\setupcapitals[..,..=..,..] **120**

title	yes no
sc	yes no

\setupcaption[...][..,..=..,..] **281**

...	*name*
..=..	see p 281: \setupcaptions

\setupcaptions[..,..=..,..] **281**

location	top bottom none high low middle
width	fit max *dimension*
headstyle	normal bold slanted boldslanted type cap small... *command*
style	normal bold slanted boldslanted type cap small... *command*
number	yes no
inbetween	*command*
align	left middle right no
conversion	numbers characters Characters romannumerals Romannumerals
way	bytext by*section*

\setupcolor[...] **143**

...	*name*

\setupcolors[..,..=..,..] **141**

state	start stop global local
conversion	yes no always
reduction	yes no
rgb	yes no
cmyk	yes no

`\setupcolumns[..,..=..,..]` **79**

n	*number*
rule	on off
height	*dimension*
tolerance	verystrict strict tolerant verytolerant stretch
distance	*dimension*
balance	yes no
align	yes no text
blank	fixed halfline line flexible big medium small
option	background
direction	left right
..=..	see p 264: \setupframed

`\setupcombinations[..,..=..,..]` **285**

before	commando
inbetween	commando
after	commando
distance	*dimension*
height	*dimension* fit
width	*dimension* fit
align	no left right middle normal

`\setupcombinedlist[...][..,..=..,..]` **190**

...	*name*
level	1 2 3 4 *section* current
..=..	see p 189: \setuplist

`\setupdescriptions[..,...,..][..,..=..,..]` **227**

...	*name*
headstyle	normal bold slanted boldslanted type cap small... *command*
style	normal bold slanted boldslanted type cap small... *command*
color	*name*
width	fit broad *dimension*
distance	*dimension*
sample	*text*
text	*text*
align	left middle right
margin	standard yes no *dimension*
location	left right top serried inmargin inleft inright hanging
hang	fit broad *number*
before	*command*
inbetween	*command*
after	*command*
indentnext	yes no

A

`\setupenumerations`*[..,...,..]*`[..,..=..,..]` **230**

...	*name*
..=..	see p 225: \definedescription
location	left right top serried inmargin inleft inright hanging
text	*text*
levels	*number*
conversion	numbers characters Characters romannumerals Romannumerals
way	bytext by*section*
blockway	yes no
sectionnumber	yes number no
separator	*text*
stopper	*text*
coupling	*name*
couplingway	global local
number	no *name*
aligntitle	no yes
start	*number*

`\setupexternalfigures[..=..]` **301**

option	frame empty test
object	yes no
frames	on off
ymax	*number*
xmax	*number*
directory	*text*
location	local global

`\setupfillinlines[..,..=..,..]` **254**

width	*dimension*
margin	*dimension*
distance	*dimension*
before	*command*
after	*command*

`\setupfillinrules[..,..=..,..]` **254**

width	fit broad *dimension*
distance	*dimension*
before	*command*
after	*command*
style	normal bold slanted boldslanted type cap small... *command*
n	*number*
interlinespace	small medium big
separator	*text*

`\setupfloat[...][..,..=..,..]` **280**

...	*name*
height	*dimension*
width	*dimension*
pageboundaries	*list*
..=..	see p 264: \setupframed

`\setupfloats[..,..=..,..]` **280**

location	left right middle
width	fit *dimension*
before	*command*
after	*command*
margin	*dimension*
spacebefore	*n**small *n**medium *n**big none
spaceafter	*n**small *n**medium *n**big none
sidespacebefore	*n**small *n**medium *n**big none
sidespaceafter	*n**small *n**medium *n**big none
ntop	*number*
nbottom	*number*
nlines	*number*
..=..	see p 264: \setupframed

`\setupfooter[...][..,..=..,..]` **97**

...	see p 97: \setupheader
..=..	see p 97: \setupheader

`\setupfootertexts[.1.][.2.][.3.]` **96**

.1.	text margin edge
.2.	*text section* date *mark* pagenumber
.3.	*text section* date *mark* pagenumber

`\setupfootnotedefinition[..,..=..,..]` **104**

..=..	see p 225: \definedescription

`\setupfootnotes[..,..=..,..]` **102**

conversion	numbers characters Characters romannumerals Romannumerals
way	bytext by*section*
location	page text columns high
rule	on off
before	*command*
after	*command*
width	*dimension*
height	*dimension*
bodyfont	5pt ... 12pt small big
style	normal bold slanted boldslanted type cap small... *command*
distance	*dimension*
columndistance	*dimension*
margindistance	*dimension*
n	*number*
numbercommand	\command#1
..=..	see p 258: \framed

\setupframed[..,..=..,..] **264**

height	fit broad *dimension*
width	fit broad *dimension*
offset	none overlay *dimension*
location	low depth
option	none empty
strut	yes no
align	no left right middle normal
bottom	*command*
top	*command*
frame	on off overlay
topframe	on off
bottomframe	on off
leftframe	on off
rightframe	on off
frameoffset	*dimension*
framedepth	*dimension*
framecorner	round rectangular
frameradius	*dimension*
framecolor	*name*
background	screen color none foreground *name*
backgroundscreen	*number*
backgroundcolor	*name*
backgroundoffset	*dimension*
backgrounddepth	*dimension*
backgroundcorner	round rectangular
backgroundradius	*dimension*
depth	*dimension*
corner	round rectangular
radius	*dimension*
empty	yes no
...	*text*

\setupframedtexts[..,..=..,..] **267**

bodyfont	5pt ... 12pt small big
style	normal bold slanted boldslanted type small... *command*
left	*command*
right	*command*
before	*command*
after	*command*
inner	*command*
margin	standard yes no
..=..	see p 264: \setupframed

`\setuphead[...][..,..=..,..]` **175**

...	*section*
style	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
numberstyle	normal bold slanted boldslanted type cap small... *command*
number	yes no
ownnumber	yes no
page	left right yes
continue	yes no
header	none empty high nomarking
before	*command*
inbetween	*command*
after	*command*
alternative	normal inmargin middle text
command	\command#1#2
numbercommand	\command#1
textcommand	\command#1
prefix	+ - *text*
placehead	yes no
incrementnumber	yes no *file*
align	left right normal broad
tolerance	verystrict strict tolerant verytolerant stretch
indentnext	yes no
file	*name*
expansion	yes command no

`\setupheader`*`[...]`*`[..,..=..,..]` **97**

...	text margin edge
state	normal stop start empty high none nomarking *name*
strut	yes no
style	normal bold slanted boldslanted type cap small... *command*
leftstyle	normal bold slanted boldslanted type cap small... *command*
rightstyle	normal bold slanted boldslanted type cap small... *command*
leftwidth	*dimension*
rightwidth	*dimension*
before	*command*
after	*command*

`\setupheadertexts`*`[.1.]`*`[.2.][.3.]` **96**

.1.	text margin edge
.2.	*text section* date *mark* pagenumber
.3.	*text section* date *mark* pagenumber

`\setupheadnumber[.1.][.2.]` **176**

.1.	*section*
.2.	*number +number -number*

\setupheads[..,..=..,..] **175**

sectionnumber	yes number no
alternative	normal margin middle text paragraph
separator	*text*
command	\command#1#2

\setupheadtext*[...]*[..=..] **164**

...	nl fr en uk de sp cz ..
name	*text*

\setuphyphenmark[..=..] **166**

sign	-- --- -) (= /

\setupindentations*[..,...,..]*[..,..=..,..] **233**

...	*name*
style	normal bold slanted boldslanted type cap small... *command*
headstyle	normal bold slanted boldslanted type cap small... *command*
width	fit *dimension*
text	*text*
sample	*text*
before	*command*
after	*command*
distance	*dimension*
separator	*text*

\setupindenting[..,...,..] **66**

...	none small medium big next first *dimension*

\setupinmargin*[...]*[..,..=..,..] **75**

...	left right *number*
location	left right both
style	normal bold slanted boldslanted type cap small... *command*
before	*command*
after	*command*
align	inner outer left right middle normal no yes
line	*number*
distance	*dimension*
separator	*text*
..=..	see p 264: \setupframed

`\setupinterlinespace[...][..,..=..,..]` **65**

...	reset small medium big on off
height	*number*
depth	*number*
line	*dimension*
top	*number*
bottom	*number*

`\setupitemize[.1.][..,.2.,..][..,..=..,..]` **237**

.1.	*number* each
.2.	standard n*broad n*serried packed unpacked stopper joinedup atmargin inmargin autointro loose section
margin	no standard *dimension*
width	*dimension*
distance	*dimension*
factor	*number*
items	*number*
before	*command*
inbetween	*command*
after	*command*
beforehead	*command*
afterhead	*command*
headstyle	normal bold slanted boldslanted type cap small... *command*
marstyle	normal bold slanted boldslanted type cap small... *command*
symstyle	normal bold slanted boldslanted type cap small... *command*
stopper	*text*
n	*number*
symbol	*number*
align	left right normal
indentnext	yes no

`\setupitems[..,..=..,..]` **245**

location	left right inmargin top bottom
symbol	1 2 ... n a ... *text* none
width	*dimension*
n	*number* unknown
before	*command*
inbetween	*command*
align	left right middle margin
after	*command*

`\setuplabeltext[...][..=..]` **164**

...	nl fr en uk de sp cz ..
name	*text*

`\setuplanguage[...][..,..=..,..]` **162**

..=..	see p 162: \installlanguage

`\setuplayout[..,..=..,..]` **32**

width	*dimension* fit middle
height	*dimension* fit middle
backspace	*dimension*
topspace	*dimension*
margin	*dimension*
leftmargin	*dimension*
rightmargin	*dimension*
header	*dimension*
footer	*dimension*
top	*dimension*
bottom	*dimension*
leftedge	*dimension*
rightedge	*dimension*
headerdistance	*dimension*
footerdistance	*dimension*
topdistance	*dimension*
bottomdistance	*dimension*
leftmargindistance	*dimension*
rightmargindistance	*dimension*
leftedgedistance	*dimension*
rightedgedistance	*dimension*
horoffset	*dimension*
veroffset	*dimension*
style	normal bold slanted boldslanted type cap small... *command*
marking	on off color
location	left middle right bottom top singlesided doublesided
scale	*dimension*
nx	*number*
ny	*number*
dx	*dimension*
dy	*dimension*
lines	*number*
grid	yes no

`\setuplinenumbering[..,..=..,..]` **91**

conversion	numbers characters Characters romannumerals Romannumerals text
start	*number*
step	*number*
width	*dimension*
location	intext inmargin
style	normal bold slanted boldslanted type cap small... *command*
prefix	*text*
referencing	on off

`\setuplines[..,..=..,..]` **90**

before	*command*
after	*command*
inbetween	*command*
indenting	yes no even odd

`\setuplist[..,...,..][..,..=..,..]` **189**

...	*name*
state	start stop
alternative	a b c ... none command
coupling	on off
criterium	*section* local previous all
pageboundaries	*list*
style	normal bold slanted boldslanted type cap small... *command*
numberstyle	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
pagestyle	normal bold slanted boldslanted type cap small... *command*
color	*name*
command	\command#1#2#3
numbercommand	\command#1
textcommand	\command#1
pagecommand	\command#1
interaction	sectionnumber text pagenumber all
before	*command*
after	*command*
inbetween	*command*
left	*text*
right	*text*
label	yes no
prefix	yes no
pagenumber	yes no
sectionnumber	yes no
aligntitle	yes no
margin	*dimension*
width	*dimension* fit
height	*dimension* fit broad
depth	*dimension* fit broad
distance	*dimension*
separator	*text*
symbol	none 1 2 3 ...
expansion	yes no command
maxwidth	*dimension*
..=..	see p 258: \framed

`\setupmakeup[...][..,..=..,..]` **108**

...	*name*
width	*dimension*
height	*dimension*
voffset	*dimension*
hoffset	*dimension*
page	left yes right
commands	*command*
doublesided	yes no empty
headerstate	normal stop start empty none nomarking
footerstate	normal stop start empty none nomarking
textstate	normal stop start empty none nomarking
topstate	stop start
bottomstate	stop start

A

`\setupmarginblocks[..,..=..,..]` **295**

location	inmargin left middle right
style	normal bold slanted boldslanted type cap small... *command*
width	*dimension*
align	left middle right no
top	*command*
inbetween	*command*
bottom	*command*
left	*command*
right	*command*
before	*command*
after	*command*

`\setupmarginrules[..=..]` **270**

level	*number*
thickness	*dimension*

`\setupmarking[...][..=..]` **206**

...	*name*
state	start stop
separator	*command*
expansion	yes no

`\setupnarrower[..,..=..,..]` **68**

left	*dimension*
right	*dimension*
middle	*dimension*

`\setupoppositeplacing[..=..]` **294**

state	start stop

`\setuppagenumbering[..,..=..,..]` **93**

alternative	singlesided doublesided
location	header footer left right middle margin marginedge inleft inright
conversion	numbers characters Characters romannumerals Romannumerals
style	normal bold slanted boldslanted type cap small... *command*
left	*text*
right	*text*
way	bytext by*section* bypart
text	*text*
numberseparator	*text*
textseparator	*text*
sectionnumber	yes no
separator	*text*
state	start stop
command	\command#1

`\setuppalet[...]` **148**

...	*name*

`\setuppapersize[..,.1.,..][..,.2.,..]` **29**

.1.	A3 A4 A5 A6 letter ... CD *name* landscape mirrored rotated 90 180 270
.2.	A3 A4 A5 A6 letter ... *name* landscape mirrored rotated negative 90 180 270

`\setupparagraphs[.1.][.2.][..,..=..,..]` **82**

.1.	*name*
.2.	*number* each
style	normal bold slanted boldslanted type cap small... *command*
width	*dimension*
height	*dimension*
align	left right middle width breedte
tolerance	verystrict strict tolerant verytolerant stretch
distance	*dimension*
before	*command*
after	*command*
inner	*command*
command	*command*
rule	on off

`\setupquote[..,..=..,..]` **247**

before	*command*
after	*command*
style	normal bold slanted boldslanted type cap small... *command*

`\setupreferencing[..=..]` **212**

state	start stop
sectionnumber	yes no
prefix	+ - *text*
interaction	label text all symbol
width	*dimension*
left	*command*
right	*command*
convertfile	yes no
separator	*text*

A

`\setupregister[.1.]`*`[.2.]`*`[..,..=..,..]` **218**

.1.	*singular name*
.2.	*name*
n	*number*
balance	yes no
align	yes no
style	normal bold slanted boldslanted type cap small... *command*
pagestyle	normal bold slanted boldslanted type cap small... *command*
textstyle	normal bold slanted boldslanted type cap small... *command*
indicator	yes no
coupling	yes no
sectionnumber	yes no
criterium	*section* local all part
distance	*dimension*
symbol	1 2 ... n a ... none
interaction	pagenumber text
expansion	yes command no
referencing	on off
command	\command#1
location	left middle right
maxwidth	*dimension*
unknownreference	empty none

`\setupscreens[..,..=..,..]` **151**

method	dot rule external
resolution	*number*
factor	*number*
screen	*number*

`\setupsection[...][..,..=..,..]` **182**

...	*name*
conversion	numbers characters Characters romannumerals Romannumerals
previousnumber	yes no

`\setupsectionblock[...][..,..=..,..]` **182**

...	*name*
number	yes no
page	yes right
before	*command*
after	*command*

`\setupsorting[...][..,..=..,..]` **203**

...	*name*
before	*command*
after	*command*
command	\command#1
state	start stop
criterium	all used
style	normal bold slanted boldslanted type cap small... *command*
expansion	yes command no

A

\setupspacing[...] **72**

...	broad packed

\setupsubpagenumber[..,..=..,..] **95**

way	bytext by*section* bypart
state	start stop none

\setupsynonyms[...][..,..=..,..] **200**

...	*name*
textstyle	normal bold slanted boldslanted type cap small... *command*
synonymstyle	normal bold slanted boldslanted type cap small... *command*
location	left right top serried inmargin inleft inright
width	*dimension*
state	start stop
criterium	all used
conversion	yes no
expansion	yes command no
command	\command#1#2#3

\setuptext*[...]*[..,..=..,..] **99**

...	see p 97: \setupheader
..=..	see p 97: \setupheader

\setuptextrules[..,..=..,..] **256**

location	left inmargin
before	*command*
after	*command*
inbetween	*command*
width	*dimension*
distance	*dimension*
bodyfont	5pt ... 12pt small big
color	*name*
style	normal bold slanted boldslanted type cap small... *command*
rulecolor	*name*

\setuptexttexts*[.1.]*[.2.][.3.] **98**

.1.	text margin edge
.2.	*text section* date *mark* pagenumber
.3.	*text section* date *mark* pagenumber

\setupthinrules[..=..] **253**

interlinespace	small medium big
n	*number*
before	*command*
after	*command*

A

\setuptolerance[..,...,..] **88**

...	horizontal vertical stretch verystrict strict tolerant verytolerant

\setuptop*[...]*[..,..=..,..] **99**

...	see p 97: \setupheader
..=..	see p 97: \setupheader

\setuptoptexts*[.1.]*[.2.][.3.] **98**

.1.	text margin edge
.2.	*text section* date *mark* pagenumber
.3.	*text section* date *mark* pagenumber

\setuptype[..=..] **124**

space	on off
option	slanted normal none
color	*name*

\setuptyping*[...]*[..,..=..,..] **123**

...	file typing *name*
space	on off
page	yes no
option	slanted normal commands color none
text	yes no
icommand	*command*
vcommand	*command*
ccommand	*command*
before	*command*
after	*command*
margin	*dimension* standard
evenmargin	*dimension*
oddmargin	*dimension*
blank	*dimension* small medium big standard halfline line
escape	/
indentnext	yes no
palet	*name* colorpretty
color	*name*

\setupunderbar[..,..=..,..] **258**

alternative	a b c
rulethickness	*dimension*

\setupwhitespace*[...]* **68**

...	none small medium big line fixed fix *dimension*

```
\showbodyfont[..,...,..]                                   117
...      see p 114: \setupbodyfont
```

```
\showbodyfontenvironment[..,...,..]                        130
...      see p 114: \setupbodyfont
```

```
\showcolor[...]                                            143
...      name
```

```
\showcolorgroup[.1.][..,.2.,..]                            149
.1.      name
.2.      horizontal vertical name value number
```

```
\showexternalfigures[..,..=..,..]                          306
alternative    a b c
```

```
\showframe[...]                                            33
...      text margin edge
```

```
\showgrid                                                  44
```

```
\showlayout                                                33
```

```
\showpalet[.1.][..,.2.,..]                                 149
.1.      name
.2.      horizontal vertical name value
```

```
\showprint[..,.1.,..][..,.2.,..][..,..=..,..]              48
..=..    see p 29: \setuppapersize
..=..    see p 29: \setuppapersize
..=..    see p 32: \setuplayout
```

```
\showsetups                                                33
```

```
\showstruts                                                73
```

```
\someline[ref]                                             214
```

A

\somewhere{.1.}{.2.}[*ref*]	**210**
.1. *text*	

\space	**73**

\startalignment ... \stopalignment*[...]*	**87**
..=.. see p 87: \setupalign	

\startbackground ... \stopbackground	**151**

\startbuffer*[...]* ... \stopbuffer	**296**
... *name*	

\startcolor[...] ... \stopcolor	**143**
... *name*	
... *text*	

\startcolumns*[..,..=..,..]* ... \stopcolumns	**79**
..=.. see p 79: \setupcolumns	

\startcombination[...] ... \stopcombination	**284**
... *n*m*	

\startcomponent ... ... \stopcomponent	**21**
... *file*	

\start*description*{...} ... \stop*description*	**227**
... *text*	

\start*enumeration* ... \stop*enumeration*	**230**

\startenvironment ... ... \stopenvironment	**21**
... *file*	

```
\startfloattext[.1.][ref]{.2.}{.3.} ... \stopfloat                    279
.1.     left right high middle low offset tall
.2.     text
.3.     text
.4.     text
```

```
\startframedtext[...][..,..=..,..] ... \stopframedtext                266
...     left right middle
..=..   see p 267: \setupframedtexts
```

```
\starthiding ... \stophiding                                          295
```

```
\startitemize[..,...,..][..,..=..,..] ... \stopitemize                242
...     a A KA n N m r R KR number continue standard n*broad n*serried packed stopper joinedup atmargin inmargin intro columns
..=..   see p 237: \setupitemize
```

```
\startlinecorrection ... \stoplinecorrection                          69
```

```
\startlinenumbering[...] ... \stoplinenumbering                       90
...     continue
```

```
\startlines ... \stoplines                                            89
```

```
\startlocalfootnotes ... \stoplocalfootnotes                          103
..=..   see p 102: \setupfootnotes
```

```
\startmarginblock ... \stopmarginblock                                294
```

```
\startmarginrule[...] ... \stopmarginrule                             270
...     number
```

```
\startnamemakeup ... \stopname                                        108
```

```
\startnarrower[..,...,..] ... \stopnarrower                           67
...     n*left n*middle n*right
```

```
\startopposite ... \stopopposite                                      294
```

A

```
\startpacked[...] ... \stoppacked                                   72
...      blank
```

```
\startparagraph ... \stopparagraph                                  82
```

```
\startpostponing ... \stoppostponing                               295
```

```
\startproduct ...  ... \stopproduct                                 21
...      file
```

```
\startproject ...  ... \stopproject                                 21
...      file
```

```
\startquotation[..,...,..] ... \stopquotation                      246
...      n*left n*middle n*right
```

```
\starttextrule[.1.]{.2.} ... \stoptextrule                         256
.1.      top bottom
.2.      text
```

```
\starttyping ... \stoptyping                                       122
```

```
\startunpacked ... \stopunpacked                                    72
```

```
\stretched{...}                                                    122
...      text
```

```
\sub[ref,..]                                                       242
```

```
\subject[ref,..]{...}                                              171
...      text
```

```
\subsection[ref,..]{...}                                           171
...      text
```

`\subsubject[ref,..]{...}` **171**

...	*text*

`\subsubsection[ref,..]{...}` **171**

...	*text*

`\subsubsubject[ref,..]{...}` **171**

...	*text*

`\switchtobodyfont[..,...,..]` **114**

...	5pt ... 12pt small big global

`\sym{...}` **242**

`\tex{...}` **126**

...	*text*

`\textreference[ref]{...}` **209**

...	*text*

`\textrule[.1.]{.2.}` **255**

.1.	top bottom
.2.	*text*

`\thinrule` **252**

`\thinrules[..=..]` **252**

..=..	see p 253: \setupthinrules

`\title[ref,..]{...}` **171**

...	*text*

`\translate[..,..=..,..]` **166**

name	*text*

\typ{...} **126**

...	*text*

\type{...} **122**

...	*text*

\typebuffer[...] **296**

...	*name*

\typefile{...} **123**

...	*file*

\underbar{...} **257**

...	*text*

\underbars{.. ... ..} **257**

...	*text*

\useblocks[..,.1.,..]*[..,.2.,..]* **287**

.1.	*name*
.2.	*name*

\useexternalfigure*[.1.]*[.2.]*[..,..=..,..]* **303**

.1.	*name*
.2.	*file*
scale	*number*
factor	max fit broad
wfactor	*number* max broad fit
hfactor	*number* max broad fit
width	*dimension*
height	*dimension*
frame	on off
preset	yes no
display	*file*
preview	yes no
repeat	yes no
object	yes no
type	eps mps pdf tif png jpg mov tex
method	eps mps pdf tif png jpg mov tex

\usereferences[..,...,..] **212**

...	*file*

`\version[...]` **26**

...	final concept temporary

`\vl[...]` **253**

...	*number*

`\whitespace` **68**

`\Word{...}` **121**

...	*text*

`\WORD{...}` **121**

...	*text*

`\wordright{...}` **88**

...	*text*

`\Words{.. ... ..}` **121**

...	*text*

`\writebetweenlist[.1.]{.2.}` **192**

.1.	*section name*

`\writetolist[.1.]{.2.}{.3.}` **192**

.1.	*section name*

`\writetoregister[.1.]`*[.2.]*`{..+.3.+..}` **219**

.1.	*singular name*
.2.	*text*
.3.	*text*

Index

The pagenumbers refer to the chapter or paragraph that describes the topic.

B

B

Commands

The pagenumbers refer to the chapter or paragraph that describes the command.

C

C

C

C

C

ConTEXt is a macropackage written in the typographic programming language TEX. It offers the user a wide range of tools to typeset documents. Although ConTEXt originally was written to facilitate the development of educational documents, it can handle all kind of simple and complex forms of documentation.

Therefore ConTEXt is used for a wide range of documents, like books of technical or more scolar nature, computer manuals, the often huge and complex quality assurance manuals and technical manuals to machines, books of encyclopedian nature, database(d) documents, and collections of documents that have common characteristics. Reuse of sources, data–abstraction and structure is typical a job for ConTEXt.

Due to the nature of TEX, ConTEXt is well suited to process documents coded in a medium–neutral way. Paper output as well as highly interactive screen documents are both supported. There is a very complete support for PDF and the related technologies.

ConTEXt is avaliable 'for free'. Apart from this reference manual, there are manuals for starters in several languages. There are also documents that describe the basic functionality. There are example styles as well as many examples of documents produced by ConTEXt.

for more information: www.pragma-ade.nl