ConTEXt Macros Top Ten

Corsair <chris.corsair@gmail.com>

Contents

1 Introduction

ConTEXt has a huge quantity of commands which might shoo away novices. For example, there are at least 9 commands to change characters' case and typeset them properly, including `\cap`, `\Cap`, `\kap`, `\Words`, etc.. So in the beginning of the ConTEXt manual, the author lists ten groups of them which are considered most frequently used:

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`

So let's walk 'em through!

2 The `text` Environment

You just cannot typeset anything if not use this environment. It's like the `document` environment in LaTeX. In ConTeXt, this (and all) environment is issued with `\start` and `\stop`.

```
\starttext
Once there was a mountain ...
\stoptext
```

Like in LaTeX, it is preferred to setup the whole document before `\starttext` so that structure and style are separated.

3 Structure of An Article

You *do* know that articles have structure, don't you? Remember when we were young and the world was in chaos, we used to typeset a line of title in M$ Word, by bold-ing, large-ing and centering it? Yes, but that was bad old time. Today we use `\section` and `\title`!

```
\starttext
  \section[sec:mnt]{The Mountain}
  Once there was a mountain ...
\stoptext
```

> **1 The Mountain**
>
> Once there was a mountain . . .

The `[...]` stuff is an optional label of this head, which enables you to cross-reference it easily with `\in`, `\at` or `\about`.

`\subject` is another unnumbered version of `\section`. Some of the structure elements are listed in table 1.

level	numbered title	unnumbered title
1	`\part`	
2	`\chapter`	`\title`
3	`\section`	`\subject`
4	`\subsection`	`\subsubject`
5	`\subsubsection`	`\subsubsubject`

Table 1 Structure Elements In ConTEXt

Now we know how to typeset heads. But what if we want to make all heads in a different font, for example, change them to sans-serif font (like in this artile)? Must we add a `\ss` in every section, subsection, subsubsection ? Definitely no. ConTEXt provides a rather convenient interface to change whatever style, including that of titles, in your article.

```
\setuphead
  [section,subsection,subsubsection]
  [numberstyle=\ssbf, textstyle=\ssbf]
```

Put this block of code before `\starttext`, and boom, we're done. In fact, in ConTEXt, almost everything can be setup in this way. Take `itemize` environment as example which is used to typeset a list, `\setupitemize[packed]` makes all lists be rendered in a compact style.

Structuring elements have a bunch of other options, which are explained in *ConTEXten* (ConTEXt manual), section 8.2.

With these structures, we can form a table of contents for out article with `\completecontent` which is similar to `\tableofcontents` in LATEX, except that the former one is much much more customizable.

4 Fancy Fonts

Why do we use different fonts in the main text? Well, we have our reasons. The most common one is to emphasize something. In ConTEXt (actually in plain TEX) we use `\em` to do it. It does various things in different environment. Normally, it toggles "slant-ish-ness" of the text, while in quoted text, it might underline it.

```
Once there was a {\em mountain},
in the {\sl mountain {\em there} was a temple},
in which a {\bf {\em monk} was telling a story}.
```

> Once there was a *mountain*, in the *mountain* there *was a temple*, in which a ***monk*** **was telling a story**.

So why bother using `\em` instead of `\sl`? Simple. `\em` tells us the text there is important while `\sl` does not. If we use `\sl` in all cases, what so important about a important block of text comparing to a book's name?

Besides italic and bold fonts, TEX make heavy use of SMALL CAP characters which are look like small upper-cased characters, though in theory they are *not* upper-cased characters in small size. TEXers love small cap fonts because they look really nice in some font families that come with their TEX distributions (and small cap is one of the many traditions in typography). In fact, small cap faces in those families are separately designed just like italic and bold faces.

Generally, small cap is used to indicate that a word is an abbreviation or an acronym. For example, "PDF" is produced from `\cap{pdf}`, and "HTML" from `\cap{html}`.

5 Lists

Lists help us thinking. There are at least two kinds of list existing in this universe: numbered and unnumbered. In LaTeX, we use `itemize` and `enumerate` environments respectively, while in ConTeXt, `itemize` rules them all.

```
\startitemize[n]
\item Atom Heart Mother
\item Dark Side of The Moon
\item Wish You Were Here
\stopitemize
```

1. Atom Heart Mother
2. Dark Side of The Moon
3. Wish You Were Here

The [n] tells ConTEXt that this list is a numbered one. Other options are listed in table 2.

option	prefix	option	prefix
n	1,2,3	1	dot
a	a,b,c	2	dash
A	A,B,C	3	star
KA	A,B,C	4	triangle
r	i,ii,iii	5	circle
R	I,II,III	6	big circle
KR	I,II,III	7	bigger circle
m	1,2,3	8	square
g	α,β,γ		
G	A,B,☒		

Table 2 Table Of All List Prefixes

Yes, every possible prefix you can think of is here for you. And if you think that space between lines is way too large, add a `packed` option to `\startitemize` or set it up in the way I discribed in section 3, page 7.

There's a `\head` macro also, to typeset fancy two-leveled lists like the following:

```
\startitemize[1,packed]
  \head Pink Floyd \par
  \startitemize[g]
  \item Atom Heart Mother
  \item Dark Side of The Moon
  \item Wish You Were Here
  \stopitemize
  \head Muse \par
  \startitemize[continue]
  \item Black Holes and Revelations
  \item Absolution
  \item Hullabaloo
  \stopitemize
\stopitemize
```

- Pink Floyd
  α. Atom Heart Mother
  β. Dark Side of The Moon
  γ. Wish You Were Here
- Muse
  δ. Black Holes and Revelations
  ε. Absolution
  ζ. Hullabaloo

6 Long Things Short

In section 4, page 9, we referred that usually people use small cap font to indicate a synonym or abbreviation. However, if you are writing a really formal document like a science paper, you might want to have more control over synonyms and even had a index of them. In ConTEXt, we can do this with `\definesynonyms` and `\completelistofabbreviations`.

First off, we define a new type of synonym called `abbr`:

```
\definesynonyms[abbr][abbrs][\infull]
```

The three arguments are all arbitrary. The first two are singular and plural name of this kind of synonym, and the third one is the command that we will use to expand the abbreviations.

Now we define an `abbr`:

```
\abbr{CSS}{Cascading Style Sheets}
```

Thus we can use this abbreviation in our article:

```
\infull{CSS} is a way of rendering the style of a web page from
multiple sources with a defined order of precedence where the
```

```
definitions of any style element conflict.
```

> Cascading Style Sheets is a way of rendering the style of a web page from multiple sources with a defined order of precedence where the definitions of any style element conflict.

The newly defined macro `\abbr` has another usage, which can define a macro to make our fingers comfortable:

```
\abbr[css]{CSS}{Cascading Style Sheets}
```

Thus, `\css` does the same things as `\infull{CSS}` does.

Also, `\definesynonyms` has another form:

```
\definesynonyms[abbr][abbrs][\inshort][\infull]
```

Then we have a new macro `\inshort` which just leave the abbreviation be.

```
\infull{CSS} (\inshort{CSS}) is a way of rendering the style
of a web page from multiple sources with a defined order of
precedence where the definitions of any style element conflict.
```

> Cascading Style Sheets (CSS) is a way of rendering the style of a web page from multiple sources with a defined order of precedence where the definitions of any style element conflict.

Like the table of contents, we can have a list of all abbrs:

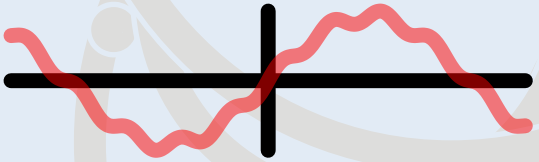
```
\completelistofabbrs
```

In case you don't know, "abbrs" in the macro above is the plural name we defined with `\definesynonyms`.

7 Figures Talk

If you have used LaTeX before, you might have noticed how hard to work with figures in it. It even needs an external macro package to do that! But in ConTeXt, we have `\externalfigure`.

```
\externalfigure[sin.pdf][width=7cm]
```



That's it. We inserted a figure into our article. Normally ConTeXt will identify same image file inserted multiple times, and reuse it to save memory and reduce output file size. If you are kind of in doubt with it and want a fine control, `\useexternalfigure` is for you.

```
\useexternalfigure[sine][sin.pdf][width=3cm]
\externalfigure[sine]
```

However, TeXers hardly like typesetting a figure in this way that the picture is fixed there where the codes are. More often do they use a "floating environment"

instead, in which position of a figure is not fixed, but, in theory, “hanging around” in a specific area, and the actual position is determined by an algorithm stored in the TEX compiler. ConTEXt provides \placefigure macro to place a floating figure.

```
\placefigure[here][fig:smile]{\Words{GNU post}}
  {\externalfigure[post.pdf][width=6cm]}
```

Figure 1 GNU Post

The [here] option tells ConTEXt that we prefer this figure right here if it is really fit. It can also be left, right, top, bottom, . . .We can as well ignore this option which leaves here as default.

Furthermore, multiple images can be placed in one block using the combination environment.

```
\placefigure
  [bottom]
  [fig:comb]
  {\Words{Multiple figures in one floating env.}}
  {\startcombination[3*2]
     {\externalfigure[post.pdf][width=2.5cm]} {a}
     {\externalfigure[fish.pdf][width=2.5cm]} {b}
     {\externalfigure[apple.pdf][width=2cm]} {c}
     {\externalfigure[gnu.pdf][width=2.5cm]} {d}
     {\externalfigure[hacker.pdf][width=2.5cm]} {e}
     {\externalfigure[ctanlion.pdf][width=2.8cm]} {f}
   \stopcombination}
```

Well, that's all for figures. Happy playing with them! BTW., you may have noticed that `[bottom]` is used in the block of code above. Pay attention to its behavior.

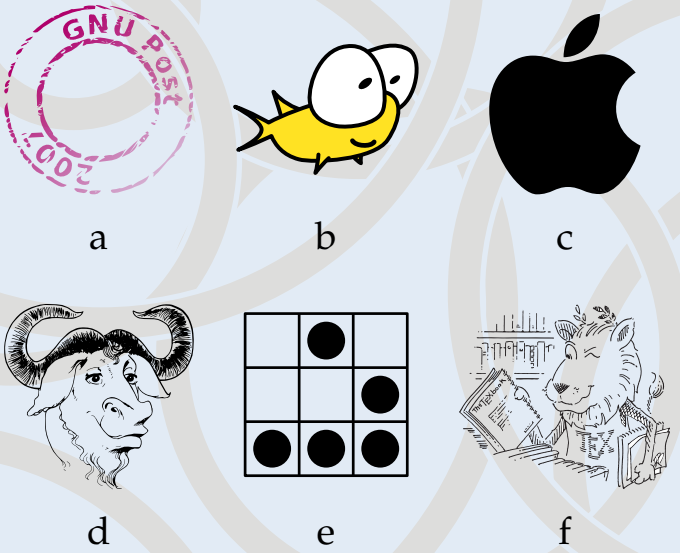


Figure 2 Multiple Figures In One Floating Env.

8 Tables

The construction of a tables in ConTEXt might be quite different with what in your mind, especially if you have used LATEX before. Let's go through an example first.

```
\starttable[| l| c| r| ]
\NC {\bf Artist} \VL {\bf Album} \NC {\bf Title} \LR
\HL
\NC Pink Floyd \NC Wish You Were Here  \VL Have A Cigar \AR
\NC Muse \NC Hullabaloo \VL Forced In \AR
\NC Queen \NC A Night at The Opera \VL Death on Two Legs \LR
\stoptable
```

Artist	Album	Title
Pink Floyd	Wish You Were Here	Have A Cigar
Muse	Hullabaloo	Forced In
Queen	A Night at The Opera	Death on Two Legs

As one can see, the formatting string which enclosed in the pair of square brace tells ConTEXt only the align of the table, though we use | to seperate the three charactors. In ConTEXt, we apply a more flexible scheme to typeset vertical lines in tables in comparison to that in LATEX that we use \VL to separate two cells so that ConTEXt will add a vertical line between them.

If you do not prefer a vertical line there, use \NC instead of \VL. All these "two-cap-lettered" macros for typesetting tables are listed in table 3. To end a row, we usually use a \AR macro or, instead, \LR if it is either the last row or followed by a \HL that draw a horizontal line across the table.

ConTEXt provides a whole bunch of macros and options for column and cell definition. See a full list of them at the ConTEXt wiki, table page.

Like a figure, a table can be a floating object whose placement is calculated by ConTEXt.

```
\placetable[right][tab:cell-def]{Cell
Definition Macros}{
  \starttable[|c|l|]
    \NC {\bf Macro} \VL {\bf Meaning}
\LR
    \HL
    \NC \type{\NC} \VL next column \AR
    \NC \type{\HL} \VL horizontal line
\AR
    \NC \type{\VL} \VL vertical line
\AR
    \NC \type{\NR} \VL next row \LR
    ...
  \stoptable}
```

Macro	Meaning
\NC	next column
\HL	horizontal line
\VL	vertical line
\NR	next row
\SR	single row
\FR	first row
\MR	middle row
\LR	last row
\AR	automatic row

Table 3 Cell Definition Macros

9 Descriptions

What you will write if you want to when you want to define something or describe stuff your readers might be interested in? Well, an "aaa: bbb" pair seems stupid enough to demonstrate that the writer is without sufficient training on typesetting. ConTEXt enables us to make it more fancy. To start with, we first use `\definedescription` macro:

```
\definedescription[definition][]
```

which defines a stuff we would like to describe in default style. Then we can describe it like this

```
\definition{Fractal}
A fractal is an object or quantity that displays self-similarity,
in a somewhat technical sense, on all scales.\par
```

Fractal A fractal is an object or quantity that displays self-similarity, in a somewhat technical sense, on all scales.

Note that the trailing `\par` or an extra newline is a must to end the description. If the description contains multiple paragraphs, we can also go for an environment:

```
\startdefinition{Vimperator}
Vimperator is a free browser add-on for Firefox, which makes
it look and behave like the Vim text editor. It has similar
key bindings, and you could call it a modal web browser, as key
bindings differ according to which mode you are in.

Vimperator was written by Martin Stubenschrott. If you appreciate
my work on Vimperator and want to encourage me working on it
more, you can either send me greetings, patches or make a
donation.
\stopdefinition
```

Vimperator Vimperator is a free browser add-on for Firefox, which makes it look and behave like the Vim text editor. It has similar key bindings, and you could call it a modal web browser, as key bindings differ according to which mode you are in.

> Vimperator was written by Martin Stubenschrott. If you appreciate my work on Vimperator and want to encourage me working on it more, you can either send me greetings, patches or make a donation.

We have already gone through lists in section 5, which is for normal vanilla itemizaion or enumerization. In this section, a more powerful technique is introduced to typeset item-structured content. The `itemize` environment provides us a good start for this kind of stuff, but what if we want a list with prefix other than table 2 has exhibited? ConTEXt surely has macros for this, one of which is `\definedescription` macro. To use this command, we may first define our own description style like this

```
\definedescription
  [myitemization]
  [location=serried,headstyle=bold,width=broad]
```

Then we have the macro `\myitemization` to typeset the description as we want:

```
\myitemization{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
```

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.

Even better, numerical index can be added if we use `\defineenumeration` to define our listing style:

```
\defineenumeration
  [remark]
  [location=top, text=Remark, between=\blank,
   before=\blank, after=\blank]
```

```
\remark The Mod Showroom is only for posting previews/downloads
of completed or near finished mods. \par
```

```
\subremark All help topics should be posted in the Editing
Discussion forums. Help topics and mod requests posted here
will be locked or binned. Thank you. \par
\remark Spaceeinstein's All In One Mod v2, A lot more mods
rolled into one. \par
```

Remark 1

The Mod Showroom is only for posting previews/downloads of completed or near finished mods.

Remark 1.1

All help topics should be posted in the Editing Discussion forums. Help topics and mod requests posted here will be locked or binned. Thank you.

Remark 2

Spaceeinstein's All In One Mod v2, A lot more mods rolled into one.

10 Build a Dictionary

. . . or not. Ok, this section is not about how to build a dictionary, though the underlying principle is no different—indexing. This is not a feature that every book needs. But when provided in a proper book, it is usually coming handy. In ConTEXt, indexing is really made easy. All you have to do is inserting a `\index` at wherever the phrase you want to index is, and placeing a `\placeindex` where you want the glossary be. For example

```
For over a thousand generations the Jedi Knights\index{Jedi}
were the guardians of peace and justice in the Old Republic\index[old
republic]{the old republic}. Before the dark times, before the
Empire\index[empire]{the empire}.
```

This renders just like a normal paragraph:

> For over a thousand generations the Jedi Knights were the guardians of peace and justice in the Old Republic. Before the dark times, before the Empire.

Note that the words in the square braces (if any) are those that are taken into account when indexing. Now let's place the glossary

```
\placeindex
```

e
the empire *29*

o
the old republic *29*

j
Jedi *29*

Like many other ConTEXt commands, users can define their own series of indexing, which pluses the default `\index` series are called register. A style of register can be defined using `\defineregister`

```
\defineregister[refer][refers]
```

in which the second argument is the plural form of the name. Then we can use `\refer` and `\placerefer` just as `\index`.

11 Page Layout

Here comes the final section, in which we will deal with some layout issue. The most import command, of course, is `\setuplayout`. I usually put one in the third line of any ConTEXt document while the first line is probably a `\usemodule` and the second is `\setuppapersize`.

Generally, there are five areas that we put text into: main text, header, footer, left margin and right margin. Note that adjusting header and footer will influence the position and height of main text, while margins will not. The dimensions of a page from top to bottom are `topspace`, `header`, `footer`, `bottomspace`. Practically, I usually set `width` and `height` to `fit` which means they are calculated automatically, and adjust the four dimensions for vertical placment, and `backspace` for horizontal position of main text, which gives the distance between the left edge of the page and that of the main text area. Margins seem to have nothing to do with it. They are adjusted independently using `*margin` and `*margindistance` in which `*` could be either `left` or `right`. The layout I use for this document is shown below.

```
\setuppapersize[S6][S6]
\setuplayout [width=fit, height=fit,
              rightmargin=1.5cm,
              leftmargin=1.5cm,
              leftmargindistance=0pt,
              rightmargindistance=0pt,
              topspace=1.8cm,
              header=0pt,
              footer=1.2cm,
              bottomspace=0.5cm,
              backspace=1.5cm,
              location=singlesided]
```

As you can see, I leave no space for header text and 1.2 cm for footer (pape number). While you are tuning layout, `\showlayout` and `\showframe` can be a *big* help.

Now let's start setting up header and footer. They are generally the same except the position, so I will focus on footer, which can be configured with `\setupfootertexts` command. This macro may take one, two, three, four or five arguments.

1. If only one argument is issued, it place the text in the middle of the page (as in this document), for example

   ```
   \setupfootertexts{pagenumber}
   ```

   This argument can also be `date`, a section name (like `chapter` or `subsection`), name of a mark, or some macros that will be substituted with texts (like `{\pagenumber of \totalnumberofpages}`).
2. If two arguments are given, the first one will be placed on the left and the second one right.
3. If three arguments are given, the first one will represent the location of the footer text, and can be one of `text`, `margin` and `edge` (outside the margins).

4. Four arguments can only be given when writing a double-sided document. They will be placed at even page left, even page right, odd page left and odd page right, respectively.
5. If five arguments are given, the first one will represent the location of the texts. Despite this, it is identical to the condition in which four arguments are issued.

12 Conclusion

ConTEXt rocks!

Oh, one more thing. I thank *kmc* on CTeX BBS and *Otared Kavian* for helping me improve this document, and *yulewang* on CTeX BBS for submitting it to CTAN.