Bijlage 29
Comparing CONTEXT and LATEX

Taco Hoekwater
CONTEXT Task Force
ntg-context@ntg.nl
taco.hoekwater@wkap.nl

abstract
Some aspects of CONTEXT and LATEX are compared: the political decisions, the offered functionality, size of the system, and relative speed.

keywords
CONTEXT, LATEX, comparison, copyright, size, speed, functionality

Over the past couple of months, CONTEXT has received a lot of attention and publicity in the TEX world. It seems like every second person I meet these days wants to learn about CONTEXT and is dying to know what the precise differences between CONTEXT and LATEX are. Here is a short article that tries to give an overview of the most user–visible differences.

Some larger categories that seem worth mentioning are the following:

- Functionality & Design
- Size
- Speed
- Politics & Support

Let's go over these differences. I am writing this mostly from the point-of-view of a person that currently uses LATEX and is trying to decide whether or not CONTEXT will be worth the trouble of going through yet another leaning curve.

Functionality & Design

There is one structural design decision that is most important: where LATEX is designed as a run-time extensible system (through packages) with lots of separate files that are loaded on startup, CONTEXT is conceived as a monolithic system, where all functionality is included into the format file. This difference has a large influence on the maintenance and development of the system.

Small extensions to the LATEX run-time system are relatively easy to implement, and as a consequence lots of people have indeed done so in the past. This is obviously an important advantage of LATEX from the viewpoint of a the 'casual user' who only has to ask a guru what the package is called that offers the desired functionality. One does not even have to know TEX to use LATEX.

But there is also a down-side: extensibility implemented this way forces the burdon of backward compatibility on the maintainers of the LATEX kernel. They cannot change the interface all of a sudden, breaking lots of existing macro code, simply because they found a better implementation of a certain problem. It is highly unlikely that packages like amsmath, fancyhdr and multicol will ever become more closely integrated with the LATEX kernel, although these packages (and a lot of other ones) could gain a lot of functionality if they would have better support in the kernel.

In CONTEXT, the situation is very much the other way around. Adding functionality is almost never simple, and even has to be approved by the system maintenance group. A more or less monolithic system needs a higher level of control to prevent 'trojan horse' code from building up in its kernel. In general, this means that extensions cannot be done by an 'average hacker': if you need functionality that is not catered for already, you probably have to wait around for a while until somebody considers it worth implementing.

On the other hand, CONTEXT does not have to stay compatible with lots of previous versions (if you want to save the older version, just make a copy of the format file), so really large improvements are a lot easier (read: might actually be implemented in a reasonable timeframe). CONTEXT is also a lot larger than the LATEX kernel is, so the functionality you ask for might indeed be present, even without you knowing.

Changing the layout

Here is an interesting difference for people that design layouts (called classes in LATEX and environments in CONTEXT): There is no low-level interface to CONTEXT. Everything is taken care of by high-level setup commands. This may sound limiting, but in practise these commands accept so much parameters that it is usually easy to get any desired effect.

For example, here are the definitions for `\section` in both systems:

In LATEX, one uses `\@startsection` to set up and define

the sectioning mechanism (examples have been taken from the MAPS definition files in both systems):

```
\def\section{\@startsection{section}{1}{\z@}
    {-1.5\baselineskip}{.5\baselineskip}{\large\bfseries}}
```

`\@startsection` takes six arguments, that specify the following things:

#1	name of the sectioning command
#2	structural level of the command
#3	indentation from left margin
#4	space above (negation blocks indentation)
#5	space below (negation gives a run-in heading)
#6	font style used

The syntax is concise, reasonably clear, and fairly flexible. But: if you want to do something that is not possible within the arguments of `\@startsection` (which is not unlikely) you have to devise your own commands. That means learning a lot about LaTeX internals and TeX macro programming, and you get only marginal support in the form of the `\@secdef` and `\addcontentsline` commands.

In CONTEXT, almost all sectioning commands are predefined, and those already defined commands accept parameters using a key–value system. Setting the parameters used while typesetting goes like this:

```
\setuphead
  [section]
  [style=\bfa,
   before={\blank[line,halfline]},
   after={\blank[halfline]}]
```

`\setuphead` has only two arguments: the sectioning command it applies to, and a list of settings. In the second argument, the following keywords are recognised (you have to guess the meanings, this article is not a manual): *style, textstyle, numberstyle, page, continue, head, before, after, command, numbercommand, textcommand, prefix, placehead, ownnumber, variant, color, distance, incrementnumber, indentnext.*

Since a lot of these arguments take TeX commands as settings, it is quite unlikely that you will need to define your own commands. It follows that in CONTEXT style design is a lot simpler than in LaTeX. In fact, lots of CONTEXT input files I have start off with twenty or thirty lines of code that setup the layout, without using any external files.

Run time loaded files

This brings us to the next difference: LaTeX typically loads somewhere between five and forty files at runtime, whereas CONTEXT usually only needs the layout and font specifications for the current document as a separate file.

This has a major impact on the structure of the filesystem, of course. LaTeX needs a tree of subdirectories to store the often needed configuration and extension files, where CONTEXT does all with only one `\input` directory, that can be virtually empty. A quick count on my harddisk showed 417 files in the LaTeX 'packages' tree, and only 17 CONTEXT run-time files (most of those are font specification files).

Functional comparison

Almost all of the functionality from the LaTeX kernel is supplied in CONTEXT. The only thing I can think of that is really missing is the `picture` environment (and one can consider that a feature: CONTEXT actively promotes the use of METAPOST instead).

Of course, a monolithic system is not as flexible as the run-time approach, but the following functionality from various LaTeX packages I am aware of (Piet van Oostrum could probably increase this list a bit) is supplied within CONTEXT (albeit usually taking a slightly different approach): *a3, a4, a5, abbrev, afterpage, alltt, amssymb, amstext, babel, bm, boxedminipage, chapterbib, changebar, color, dcolumn, doc, doublespace, draftcopy, endfloat, endnotes, enumerate, example, exscale, fancybox, fancyhdr, fancyvrb, flafter, fleqn, float, fnpara, fontenc, footnote, ftnright, graphic*, graphpap, here, hhline, hyperref, ifthen, indentfirst, inputenc, keyval, lablst, layout, leqno, letterspace, longtable, lscape, ltxdoc, lucida*, makeindx, marks, metapost, mfnfss, minitoc, moresizes, moreverb, multicol, multidx, multirow, picins, picinpar, psnfss, samepage, showkeys, sidefloat, slides, syntonly, sub*, tabularx, testpage, theorem, trace*, verbatim, vpage, wrapfig, xspace*

There are some things, however, that are not (yet) implemented. I consider the following to be the most important packages, but the same remark as above applies to this list as well: *amsmath, array, bib*, breqn, natbib.*

And then there is some extra functionality that doesn't have a parallel in LaTeX as far as I know: the multilingual interface, interactive menu's, syntax highlighted verbatims, selective processing, run–time generated METAPOST drawings, reader profiles, collated and positioned output, two-pass page break/float optimalization, synonyms/glossaries, 'real' plain TeX compatibility, text buffers, page and paragraph backgrounds, color palettes, extended list item support, parallel documents, floats in multiple column output and finally grid snapping.

Size

Measurements of the relative run-time sizes of the two packages can be done on a couple of different levels. First and foremost, here are the TEX capacities needed to process a trivial 'Hello World' file in both systems. The files used to obtain the numbers are given below:

For LATEX:

```
\documentclass{minimal}
\begin{document}
Hello, World!
\end{document}
```

For CONTEXT:

```
Hello, world!
\bye
```

```
Here is how much of TeX's memory you used:
 13 strings
 191 string characters
 541649 words of memory
 2948 multiletter control sequences
 3640 words of font info for 14 fonts
 14 hyphenation exceptions
 14i,4n,10p,107b,137s stack positions
```

Figure 1 Parameter usage for LATEX.

```
Here is how much of TeX's memory you used:
 210 strings
 2081 string characters
 718520 words of memory
 16358 multiletter control sequences
 9049 words of font info for 28 fonts
 15 hyphenation exceptions
 50i,16n,65p,85b,898s stack positions
```

Figure 2 Parameter usage for CONTEXT.

The LATEX version resulted in the log file given in figure 1, the CONTEXT version gave figure 2. It is interesting to note that CONTEXT doesn't use all that much strings yet, although it almost exclusively uses a key–value system for options. On the other hand, CONTEXT uses a *lot* of control sequences, and puts quite a strain on the 'save stack' (that's the `898s`). Remember these are values for a minimal file, for a full production document the CONTEXT values will look more like those given in figure 3.

```
Here is how much of TeX's memory you used:
 1559 strings
 16366 string characters
 744952 words of memory
 17674 multiletter control sequences
 52309 words of font info for 104 fonts
 15 hyphenation exceptions
 53i,17n,91p,127b,1731s stack positions
```

Figure 3

What we see here is that apparently there are quite some macros that either create new strings or define other control sequences (since these definately have not been created by the loading of extra macro files).

Both the hash table size and the save stack are easily flooded in old–fashioned systems, but with the advent of web2c 7.0 it luckily became very easy to change the values of TEX's various memory paramaters. CONTEXT definately needs a *large* TEX. The values I use are given below (some of those parameters are set high because of other things I do, not for CONTEXT):

```
main_memory   =1500000 % words of inimemory
extra_mem_top = 500000 % extra high memory
extra_mem_bot = 500000 % extra low memory
font_mem_size = 200000 % Words of font info
font_max      =   1000 % Total number of fonts
hash_extra    =  30000 % Extra Hash table entries
pool_size     = 500000 % String values
string_vacancies=50000
max_strings   =  25000
pool_free     =  10000
trie_size     =  64000 % hyphenation trie
hyph_size     =   1000 % hyphenation exceptions
nest_size     =    200 % semantic groups
max_in_open   =     40 % input files
param_size    =   1000 % macro parameters
save_size     =  10000 % for saving values
stack_size    =    600 % input sources
```

Some other size numbers are the ones that are related to the size of the macro packages on disk, both the format file and the format's sources and support macros.

The typical CONTEXT format file is slightly over 2.3 Megabytes, compared to about 550 Kilobytes for LATEX. This 5 : 1 ratio reflects itself in the sources as well. Comparing the disk occupation roughly gives the

same ratio for the base system (not counting LaTeX's various 'standard' packages), and indeed the printed sources of ConTeXt are about five times as large a pile of paper as the LaTeX pile (you are not advised to try this unless you have a *lot* of paper and a very fast printer available).

Speed

Now let's talk about speed. ConTeXt offers quite a lot of functionality that is not available in LaTeX, and it is completely parameter–driven, so it seems quite reasonable to expect that it will be a bit slower. But how much is a bit?

Here are some timings, done with a very plain input file that creates 200 pages of impressively boring text. The ConTeXt file has been processed three times, to show the relative time spent in the output routine. For LaTeX, this did not seem to make much sense, since LaTeX was rather fast already and also because setting up LaTeX to use 6 point body fonts is not all that trivial.

System	Pointsize	Pages	Time
plain	10pt	181	21s
LaTeX	10pt (default)	244	27s
ConTeXt	12pt (default)	259	02m59s
ConTeXt	10pt	185	02m17s
ConTeXt	6pt	83	01m22s

We see that ConTeXt is indeed quite a lot slower. The second and third ConTeXt runs indicate that it spends a far larger amount of time in the output routine. (This test assumes that TeX's paragraph processing will not have a large impact on the timings. Because the DVI files stay roughly the same size, file I/O can be ignored as well. What is left over is mostly time spent in `\output`.)

It follows that the output routine of ConTeXt is a lot more complicated than LaTeX's (not really a surprise) and far less optimized to deal with trivial cases like the demonstration file (which is an interesting observation). Further tests showed that ConTeXt's version of `\reset@font` (which is called `\restoreglobalbodyfont`, by the way) really takes a lot of time. Disabling this macro gave a timing of 1m50s for the 10 point version, a speed gain of almost 30 seconds![1]

I decided to run the trivial file from the previous example with `\tracingmacros=1` to see what would happen, and indeed: whereas the LaTeX version resulted in a log file of only 72 kilobytes, the ConTeXt log became 1.3 megabytes! Interestingly, ConTeXt's `\output` specifications in the sources are almost trivial, and after reading them only a couple of times you actually understand what the output routine is doing.

An impressive part of the ConTeXt sources that shows off the amazing amount of structure is the following piece of code that comes from the end of the output routine:

```
\addpagecutmarks  0
\replicatepagebox 0
\scalepagebox     0
\mirrorpaperbox   0
\rotatepaperbox   0
\centerpagebox    0
\mirrorprintbox   0
\rotateprintbox   0
\offsetprintbox   0
\pagegoal=\dimen0
\box0{}}
```

This shows the next major difference between LaTeX and ConTeXt that I consider worth noting: LaTeX is optimized for speed of processing, whereas large portions of ConTeXt are optimized for ease of comprehension (even a lot of the optimized macros contain a literate programming comment that shows the non–optimized version of the macro and explains the algorithm used, including why and how the macro was optimized).

Politics & Support

There seems to be some confusion amongst people with regard to the copyright issues and politics of both packages. The following paragraphs give the 'definitive' answers for LaTeX and ConTeXt respectively.

Usage restrictions

The LaTeX2e copyright (`legal.txt`) says absolutely nothing about usage restrictions. The result is that legally there are no such restrictions, and LaTeX is effectively *free software.* However, donations to the LaTeX3 project are actively encouraged. The file `ltx3info.tex` says:

> "Although LaTeX may be distributed freely, the production and maintenance of the system does require expenditure of reasonably large sums of money. The LaTeX3 Project Fund has therefore been set up to channel money into this work. We know that some users are aware of this fund as they have already contributed to it—many thanks to all of them! If you want to know more about how you can help the project, see Page [..]—and thanks in advance for your generosity in the future."

1 This is no longer true in the latest beta version of ConTeXt, most because Hans proof–read this article for me.

There *are* usage restrictions on CONTEXT, however. Every file in the CONTEXT distribution begins with the following text:

> "This module is part of the CONTEXT macro–package and is therefore copyrighted by PRAGMA. Non–commercial use is granted."

This is a little bit on the terse side, but a more verbose official statement is in the readme file:

> "[On public use:] The most recent stable version of CONTEXT is available in the public domain and may be used by everyone, except by direct competitors of PRAGMA ADE.
>
> [On commercial use:] Systematic large scale commercial use of CONTEXT is permitted, given permission by PRAGMA ADE, conforming the conditions as stated below.
>
> With commercial use, we mean systematic document processing for third parties as well as large scale in–company use. Using CONTEXT in an iterative process towards an optimal document (for instance an article) is of course permitted.
>
> [...]
>
> Commercial users ... should pay a decent contribution."

It should be clear that you can use CONTEXT freely if you are a private person or non–commercial institution. Hans Hagen explained the situation to me as follows (the words are as I remember them, so please don't cite me):

> "We (PRAGMA) have spent a lot of time developing CONTEXT. We are a rather small company, and this has been a major investment for us. We are quite happy to offer our macros to the general public, but we would hate to find ourselves in the situation where we actually loose income because customers or competitors start using CONTEXT themselves.
>
> If you think you might belong in one of those two categories, you should contact PRAGMA for licensing information."

Some further discussion with Hans resulted in the following two lists.

Here are some groups of users that are expressly allowed to use CONTEXT without contacting PRAGMA:

1. Companies that have to work with unsolicited author–supplied CONTEXT input files.
2. Typesetting companies (small or large) that use CONTEXT internally, but only on an incidental basis and that do not advertise this fact.
3. Non-profit organisations.
4. In all other cases, barring those enumerated below: if the total annual production of pages is less than 2.000, it is safe to presume that you belong to this group.

And here are some groups that are expressly forbidden to use CONTEXT without prior written consent by PRAGMA:

1. Publishing houses.
2. Typesetting companies that intend to advertise with the fact that they use and/or accept CONTEXT input.
3. Typesetting companies that intend to use CONTEXT as their base means of production.
4. Governmental institutions.

Distribution restrictions

Both systems share restrictions on distribution of the system and distribution of changed files. The following text has been adopted from `legal.txt` from the LATEX distribution (with some exceptions deleted, this in *not* the legal version!).

> Redistribution of unchanged files is allowed provided that all files that belong to the system are distributed.
>
> The distribution of changed versions of certain specified files (most notably, the font definition files) included in the system are allowed under the following restrictions:
>
> - You rename the file before you make any changes to it. Any such changed files should be distributed under conditions that ensure that those files, and any files derived from them, will never be redistributed under the names used by the original files in the distribution.
> - You change the 'identification string' to clearly indicate that the file is not part of the standard system.
> - You change the 'error report address' so that we do not get error reports for files not maintained by us.
> - You acknowledge the source and authorship of the original version in the modified file.
> - You also distribute the unmodified version of the file.
>
> The above restrictions are not intended to prohibit, and hence do not apply to, the updating, by any method, of a file so that it becomes identical to the latest version of that file in the Standard system.

I have the distinct impression that this is actually more restrictive than most people seem to think it is (I know it was a lot more restrictive than I myself thought it was). Of course, the restrictions serve only two purposes: protection of the rights of people that have written the original work, and protection of the integrity of LaTeX as a system.

CONTEXT has the second problem in an even higher exponent: since almost all source files are included into the format, there is no telling what would happen if their contents cannot be guaranteed. To make sure CONTEXT will stay consistent without requiring a huge amount of maintenance, there is a very simple rule: You are *not* allowed to distribute changed or derived files at all, and distribution is only allowed under control of one of the TeX User Groups.

Support & Availability

Support for LaTeX is typically handled by the various user groups (for dutch users: the `tex-nl` mailing list. For subscription information, see elsewhere in this MAPS), and globally on the `comp.text.tex` newsgroup. Apart from these, there is a separate address for bug reports. See `bugs.txt` from the distribution for more information.

All support for CONTEXT is handled by the 'CONTEXT Task Force', a group of 'pioneer users' from different countries that serve as a filter between users and PRAGMA.

This group of people monitors a mailing list kindly provided by the Dutch TeX User Group: `ntg-context@ntg.nl`. Subscription requests should be sent to `majordomo@ntg.nl` with body "subscribe ntg-context". The mailing list can also be used for bug reports and feature requests.

LaTeX and the various contributed files are available from CTAN, and the base system together with lots of packages is currently part of (almost?) all TeX distributions. CONTEXT is distributed from a central location: `http://www.ntg.nl/context`. Soon, the sources will also be uploaded to CTAN, and CONTEXT is quickly becoming part of the current TeX distributions.

A final word

It should be clear that there are some major differences between the two packages. In fact, maybe the only thing that they *do* have in common is that they are both large, independantly developed TeX macro packages.

They may appeal to completely different types of users: Creating text books and manuals (including interactive ones) with CONTEXT is surprisingly easy, but LaTeX can handle highly demanding scientific articles a little better and faster.