ConTEXt basics for users: conditional processing

Aditya Mahajan

Abstract

Many-a-times, you want to to generate multiple versions of the same document. One version for printing and one for viewing on the screen; one version for students and one version for the instructor; and so on. You can do this a naive way. Create different files with setup for the different versions and `\input` the common material, or create some new conditional flags using `\newif` and set them appropriately for conditional processing. Or you could use *modes*—the ConTEXt way of doing conditional processing.

1 Introduction

A mode is similar to a conditional flag, but with a few advantages. New modes need not be explicitly defined (no need for something like `\newif`), multiple modes can be simultaneously enabled or disabled, the status of multiple modes can be checked easily. Moreover, modes can be set from a command line switch. So, multiple versions of a document can be generated without changing the source file.

The name or identifier of a mode can be any combination of letters, digits, or spaces. Names starting with `*` are reserved for system modes.

In this article I explain how to activate a mode and how to check if a mode is active or not.

2 Setting modes

ConTEXt has three commands for setting modes:

- `\enablemode [...]`
- `\disablemode[...]`
- `\preventmode[...]`

The names are self explanatory. `\enablemode` activates a mode, `\disablemode` deactivates a mode, and `\preventmode` permanently deactivates a mode. All three commands take a list of modes as an argument. For example, you can activate `screen` and `solution` modes by

```
\enablemode[screen,solution]
```

Modes can also be activated by a command line switch `--modes` to `texexec` and `context`. For example, to activate `screen` and `solution` modes, you can run ConTEXt using

```
texexec --mode=screen,solution ...
```

or

```
context --mode=screen,solution ...
```

3 Conditional processing based on modes

Suppose you want to change the paper size of a document depending on whether it is for print or screen. This can be done in multiple ways. You could either set a default paper size for print and change it for screen:

```
\setuppapersize[letter][letter]

\startmode[screen]
  \setuppapersize[S6][S6]
\stopmode
```

(S6 is one of the screen-optimized paper sizes in ConTEXt; the paper size has a 4:3 aspect ratio and a width equal to the width of A4 paper.) Alternatively, you could set a default paper size for screen and change it if the screen mode is not enabled:

```
\setuppapersize[S6][S6]

\startnotmode[screen]
  \setuppapersize[letter][letter]
\stopnotmode
```

`\startmode` and `\startnotmode` can check for multiple modes. The arguments to `\startmode` and `\startnotmode` can be a list of modes. `\startmode` processes its contents (everything until the next `\stopmode`[1]) if any of the modes are enabled, otherwise (i.e., when all the modes are disabled) `\startmode` ignores it contents. `\startnotmode` is the opposite. It processes its contents (everything until the next `\stopnotmode` if any of the modes are disabled, otherwise (i.e., when all the mods are enabled) `\startnotmode` ignores its contents.

`\startmode` and `\startnotmode` are *or* environments. They process their contents if any of the modes satisfy the required condition. Their *and* counterparts are `\startallmodes` and `\startnotallmodes`, which process their contents only if all the modes satisfy the required condition. For example, suppose you want to enable interaction (hyperlinks) etc. only when both `screen` and `solution` modes are enabled. Then you can use:

```
\startallmodes[screen,solution]
  \setupinteraction[state=start]
\stopallmodes
```

To summarize, the four start-stop environments for checking modes are:

```
\startmode[mode1, mode2, ...]
  % Process if one of the modes is enabled
\stopmode

\startnotmode[mode1, mode2, ...]
```

[1] This means that `\startmode` cannot be nested.

```
  % Process if one of the modes is disabled
\stopnotmode

\startallmodes[mode1, mode2, ...]
  % Process if all modes are enabled
\stopallmodes

\startnotallmodes[mode1, mode2, ...]
  % Process if all the modes are disabled
\stopnotallmodes
```

These environments have a `\doif` alternative that are useful for short setups and can also be nested.

```
\doifmode        {modes} {content}
\doifnotmode     {modes} {content}
\doifallmodes    {modes} {content}
\doifnotallmodes {modes} {content}
```

The logic for determining when the content is processes is exactly the same as for the `start-stop` alternatives.

These `\doif` commands have a variant that can process something else if the conditions are not satisfied (like the `\else` branch of `\if`).

```
\doifmodeelse    {modes} {content} {alternative}
\doifnotmodeelse {modes} {content} {alternative}
\doifallmodeselse{modes} {content} {alternative}
\doifnotallmodeselse{modes}
                         {content} {alternative}
```

4 System modes

In addition to user-defined modes, ConTEXt provides some system modes. These modes start with a `*`. Here I will only explain the more commonly used system modes. You can see the ConTEXt modes manual[2] for a complete list of system modes.

Perhaps the most useful system modes are `*mkii` and `*mkiv` which determine whether MkII or MkIVis being used. These modes are handy when you want different setups for MkII and MkIV.

Other modes are useful for very specific situations. Some of these are described below.

A document multiple times to get the cross referencing, table of contents, etc. right. However, sometimes you need to do some external processing (like graphic conversion) that only needs to be done only once. In such cases, the `*first` mode is handy, which checks for the first run of the document.

You can use the project-product-component structure for managing large projects like a book series. See the ConTEXt wiki[3] for details. Both product and components can be compiled separately. Sometimes you want to process a component differently depending on whether it is being compiled directly, or if the complete product is being compiled. This can be checked using the modes `*project`, `*product`, `*component`, and `*environment`. For example, the `*product` mode is set whenever a product file is read (more specifically, whenever a `\startproduct` is encountered). Similarly, a mode `*text` is enabled whenever a `\starttext` is encountered.

A large document is also broken down into different section blocks like frontmatter, bodymatter, appendices, and backmatter. Internally, these section blocks are refered as `frontpart`, `backpart`, `appendix`, and `backmatter`. Each section block sets a system mode with the same name. So, if you want macros that work differently for different section blocks, you can check for `*fronpart`, `*backpart`, `*appendix`, and `*backmatter` modes.

ConTEXt provides support for multiple language. Language are recognized by their IETF language tags, like `en-us` for US English, `en-gb` for British English, `nl` for Dutch, `de` for German, etc. A document has a main language set using `\mainlanguage[...]` that is sued for translated labels like *chapter* and *figure*. You can also switch the current language using `\language[...]` to change the hyphenation rules. Whenever a language is chosen, its id is set as mode. The mode for the main language starts with two `*`. For example, when the main language is US English and the current language is Dutch, the modes `**en-us` and `*nl` are set (notice the extra `*` in `**en-us`).

Other system modes are: `*figure` which is set when a graphic is found, `*interaction` which is set when interaction is enabled, `*grid` which is set when grid typesetting is enabled, and `*pdf` and `*dvi` which are set depending on the whether we are generating PDF or DVI output. Others are too esoteric to desribe here. If you are interested, see the modes manual[2].

◇ Aditya Mahajan
adityam (at) umich dot edu

[2] http://pragma-ade.com/general/manuals/mmodes.pdf
[3] http://wiki.contextgarden.net/Project_structure