A quickstart guide to arara

Welcome to arara, the cool TEX automation tool! This document is intended to be a quickstart guide, providing the basic instructions to start using our tool. We will help our good friend Peter, [1] the anteater, in his learning adventure. He is very excited about it! Hello, Peter!

Peter: *Hello, nice to meet you, person from the Island of TEX ! I am Peter, the anteater. So I heard about this tool named arara, what is it? Is there anything to do with TEX and friends?*

As a matter of fact, it is! arara is a TEX automation tool. The name was chosen as an homage to a Brazilian bird of the same name, which is a macaw. The word *arara* comes from the Tupian word *a'rara* , which means *big bird* . The tool is an effort to provide a concise way to automate the daily TEX workflow for users and also package writers.

Peter: *It definitely sounds intriguing! So tell me, how does arara work? Does it do everything automatically for me, like other tools? Do things happen by magic?*

A very good question, Peter. The best way to explain how arara works is to provide a quick comparison with similar tools. I believe you are familiar with TEX documents, right? Let us use the following file `article.tex` as an example:

```
\documentclass{article}

\begin{document}
Hello world!
\end{document}
```

How would one successfully compile `article.tex` with `latexmk` and `rubber` , for instance? It is quite straightforward: it is just a matter of providing the file to the tool and letting it do the hard work:

```
$ latexmk -pdf article.tex
$ rubber --pdf article.tex
```

The mentioned tools perform an analysis on the file and decide what has to be done. However, if one tries to invoke `arara` on `article.tex` , I am afraid *nothing* will be generated; the truth is, arara does not know what to do with your file, and the tool will even raise an error message complaining about this issue:

```
$ arara article.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "article.tex" (size: 70 B, last modified: 12/28/2020
07:03:16), please wait.

ERROR

It looks like no directives were found in the provided file. Make
sure to include at least one directive and try again.

Total: 0.04 seconds
```

Peter, this is the major difference of arara when compared to other tools: *it is not an automatic process and the tool does not employ any guesswork on its own* . You are in control of your documents; arara will not do anything unless you *teach it how to do a task and explicitly tell it to execute the task* . So, it is not magic.

Peter: *I see. On the other hand, this approach gives me, as user, full control of my document compilation. Being aware of the compilation steps makes me understand the TEX tooling better. Quite clever! Now, do tell me: how does one tell arara to do a task?*

First of all, Peter, arara has to know *how* to do a task. This is done by defining rules. A rule is a formal description of how the tool should handle a certain task. For example, if we want to use `pdflatex` with arara, we need a rule for that. Once a rule is defined, the tool automatically provides an access layer to the user. Note that arara is distributed with dozens of predefined rules, so you already have several options out of the box to set up your workflow.

> **Anteater facts:**
> Giant anteaters can be found throughout South and Central America, though their numbers have diminished considerably from the latter. To thrive, they need to be able to move throughout large areas with patches of forest. They can often be found in tropical and dry forests, savannas, and open grasslands, where the ants upon which they feed are abundant. READ MORE →

Now, back to your question. Once we know how to execute a task, we need to explicitly tell arara when to do it. That is actually the easy part, provided that you have everything up and running. We accomplish the task by adding a special comment line, hereafter known as a *directive* , somewhere in our document to indicate how the tool should work, e.g:

```
% arara: pdflatex
```

Peter: *Makes sense. So, a directive is a special comment line which is not the command to be executed, but the name of the rule associated with that directive. Is that correct?*

Perfect, Peter! That is basically how arara works: we teach the tool to do a task by providing a rule, and tell it to execute it via directives in the source code. Let us see how our `article.tex` file should be:

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

\begin{document}
Hello world!
\end{document}
```

Note, Peter, that the tool expects *you* to provide a list of tasks, and this is done by inserting special comments, i.e, directives, in the source file. Since we just added one directive in our document, let us see how arara behaves with this updated code:

```
$ arara article.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "article.tex" (size: 88 B, last modified: 12/28/2020
07:05:05), please wait.

(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS

Total: 0.56 seconds
```

Peter: *Goodness me, it worked like a charm! The moment we specified a directive, arara knew exactly what I wanted to do with that particular file! Awesome!*

Exactly! You see, Peter, there is no guesswork from arara. The tool will do exactly what you tell it to do, no more, no less. There is a lot of freedom to this design, which gives you an interesting way to enhance your TEX experience.

> **Anteater facts:**
> Anteaters are not aggressive, but they can be fierce. A cornered anteater will rear up on its hind legs, using its tail for balance, and lash out with dangerous

> claws. The giant anteater's claws are some four inches long, and the animal can fight off even a puma or jaguar. READ MORE →

Peter: *I have been wondering: there are scenarios in which we need to provide additional information to the underlying commands – for instance, we need to enable shell escape when using the `minted` package. How can we achieve this?*

For such scenarios, arara provides a second type of directive, a parametrized one, which allows passing arguments to the corresponding rule. From your example, to enable shell escape, one simply needs to write the following directive:

```
% arara: pdflatex: { shell: yes }
```

Of course, `shell` is not taken randomly, but defined in the rule scope, otherwise arara would raise an error about an invalid key. The reference manual has a list of all available keys for each predefined rule. It is worth a read.

> **A useful tip**
> Do you know that a directive can be split into multiple lines? Simply use the `arara: -->` mark to each line which should compose the directive. Although it is possible to spread lines of a multiline directive all over the code, it is considered good practice to keep them together for easier reading and editing.

Peter: *Great, these directives are really convenient! I am now curious on how to explore arara: shall we move to a more complex document? Consider the following addition to my document:*

```
\documentclass{article}

\begin{document}
\section{Termite recipes}
\label{sec:termiterecipes}

Hello, this is Section~\ref{sec:termiterecipes}.
\end{document}
```

As we can see, this document has to be compiled twice, or the reference to the first section will not be resolved accordingly. How can I tackle this scenario with arara?

The solution is quite straightforward, Peter: how about adding two directives into your document? You can keep them together for convenience (usually at the top), but they can happen anywhere in your file. Let us update the code:

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

\begin{document}
\section{Termite recipes}
\label{sec:termiterecipes}

Hello, this is Section~\ref{sec:termiterecipes}.
\end{document}
```

The execution workflow is now as expected: our document was correctly compiled twice, as it should be, so references are resolved accordingly!

```
$ arara article.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing 'article.tex' (size: 196 B, last modified: 01/07/2021
08:18:00), please wait.

(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS
(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS

Total: 1.60 seconds
```

Peter: *Cool! But I have been wondering: once the references are resolved, subsequents runs will add an extra, unnecessary compilation. It would not do any harm, surely, but is there a way to avoid it?*

There is a way, Peter! We can use logical expressions and special operators and methods processed at runtime in order to determine whether and how a directive should be processed. This feature is named *directive conditional* , or simply *conditional* for short. Let us update the document to include this feature:

```
% arara: pdflatex
% arara: pdflatex if found('log', 'undefined references')
\documentclass{article}

\begin{document}
\section{Termite recipes}
\label{sec:termiterecipes}
```

```
Hello, this is Section~\ref{sec:termiterecipes}.
\end{document}
```

Observe, Peter, that the first directive has no conditional, so it will always be processed. The second one, however, has an associated logical test: check if the log file contains a warning about undefined references and, if so, process the directive itself. When references are resolved accordingly, the tool will not process the second directive at all, as we can see in the following run:

```
$ arara article.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing 'article.tex' (size: 236 B, last modified: 01/07/2021
08:42:02), please wait.

(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS

Total: 0.97 seconds
```

In this particular case, the test is evaluated beforehand, and the directive is processed if, and only if, the result of such evaluation is true. This directive, when the conditional holds true, is executed at most once.

> **A useful tip**
> When using certain conditional operators, there are no conceptual guarantees for proper halting of unbounded loops. However, do not worry! The team has provided a technical solution for potentially infinite iterations: arara has a predefined maximum number of loops. The default value is set to 10, but it can be overridden.

Peter: *These conditionals can help me dictate how my workflow should behave in certain scenarios! I gather there are several possible ways of tackling them, right?*

Correct, Peter. In the previous example, we used the `if` operator, but we could have used `while` or `until` as well. The logical expression could also be rewritten. We could even have combined these two directives into one! So you see, there is always room for improvement.

> **Anteater facts:**
> The giant anteater uses its sharp claws to tear an opening into an anthill and put its long snout, sticky saliva, and efficient tongue to work. But it has to eat quickly, flicking its tongue up to 150 times per minute. Ants fight back with

> painful stings, so an anteater may spend only a minute feasting on each mound.
> READ MORE →

If this subject caught your attention, take a look at the reference manual for more details on directive conditionals and available methods. There are multiple code excerpts to help you understand better the inner workings.

Peter: *I will certainly check them out! Now, I am curious to see other directives in action together, so let us try a different yet common scenario: bibliographies and citations. Consider the following bibliography file, containing a reference* [2] *to my doctoral thesis:*

```
@phdthesis{peter:2020,
  author  = {Peter Anteater},
  title   = {On flexibility: \LaTeX, latex and rubber},
  school  = {Polytechnic University of P{\^{a}}ntano Fundo},
  year    = 2020,
  address = {Pantanal},
  month   = {jan}
}
```

I want to cite my thesis in the article you and I are writing for this quickstart guide. From the TEX side, it is quite straightforward! How about arara? Here is my code:

```
\documentclass{article}

\begin{document}
\section{Termite recipes}
\label{sec:termiterecipes}

A citation from my thesis~\cite{peter:2020}.

\bibliographystyle{alpha}
\bibliography{bibliography}
\end{document}
```

Peter, remember that arara does not employ any guesswork, so do not expect it to magically compile your document and get all the citations correctly. We need to explicitly tell it what to do, as if we were explaining the compilation steps to a friend. arara is focused on reproducibility.

> **Anteater facts:**
> Anteaters are edentate animals — they have no teeth. But their long tongues are more than sufficient to lap up the 35,000 ants and termites they swallow whole

> each day. As the largest of all four anteater species, the giant anteater can reach eight feet long from the tip of its snout to the end of its tail. READ MORE →

We, as users, can learn a great deal about TEX and friends by organising our workflow into directives. Let us review what should be done in order to correctly compile your document:

1. For starters, we need to run the TEX engine in order to write (amongst other things) the relevant bibliography information to the auxiliary file. We can achieve this by inserting the following directive:

   ```
   % arara: pdflatex
   ```

2. Once the auxiliary file holds the relevant bibliography information, we need to run the Bib TEX tool as a means to map entries from the bibliography database to the existing citations into an intermediate bibliography file. We can achieve this by inserting the following directive:

   ```
   % arara: bibtex
   ```

3. Once the intermediate bibliography file is generated, we can now have a proper bibliography section in our document, so we need to run the TEX engine again. We can achieve this by inserting the following directive:

   ```
   % arara: pdflatex
   ```

4. However, the citations are still not yet referenced in the document, so a final run of the TE X engine is required in order to correctly adjust these references. We can achieve this by inserting the following directive:

   ```
   % arara: pdflatex
   ```

And we are done! These are the compilation steps required to correctly generate your article from the given source and bibliography files. Note that the entire workflow involves two different tools working together: the TEX engine and the Bib TEX tool.

Peter: *Wait a minute, no less than four compilation steps? I would never guessed it! This is great: arara is also helping me understand better how the TEX workflow works!*

Quite true, Peter! arara tries its best to help users think out of the box! Now that we know the exact compilation steps to be taken, we just need to arrange them in the correct order in our document:

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

\begin{document}
\section{Termite recipes}
\label{sec:termiterecipes}

A citation from my thesis~\cite{peter:2020}.

\bibliographystyle{alpha}
\bibliography{bibliography}
\end{document}
```

Now, let us take arara into action! By running the tool on our article, we can see all compilation steps being performed in the exact order we specified in the source code, as expected:

```
$ arara article.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing 'article.tex' (size: 281 B, last modified: 01/08/2021
06:17:20), please wait.

(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS
(BibTeX) The BibTeX reference management software ....... SUCCESS
(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS
(PDFLaTeX) PDFLaTeX engine .............................. SUCCESS

Total: 2.49 seconds
```

Peter: *Great! I see arara is quite expressive! I am very curious: what about other possibilities, will I learn them by looking at the reference manual? Surely there are way more features to discover.*

Of course, Peter! The reference manual contains everything there is to know about arara, so it is a great resource for learning! In this quickstart guide, we simply covered the basics, as an introductory material. The tool has several features, including support for working directory, processing of multiple files, argument passing through command line flags, configuration files, default preambles, file hashing, safe mode, and much more. There is a world of possibilities!

Peter: *Thank you very much, person from the Island of TEX ! I am sure I will have a lot of fun with arara! Should I need any assistance, how can I contact the team?*

Great talking to you, Peter! If you run into any issue with arara, please let us know. We all have very active profiles in the TEX COMMUNITY AT STACKEXCHANGE , so just use the `arara` tag in your question and we will help you the best we can (also, take a look at their STARTER GUIDE). We also have GITTER and MATRIX chat rooms, in which we occasionally hang out. Also, if you think the report is worthy of an issue, open one in our GITLAB REPOSITORY . Happy TEX ing with arara, Peter!

1

Peter was graduated from Termite High School, Alta Floresta, Mato Grosso, Brazil. He went on to study at the California Institute of TikZlings with a scholarship from San Diego Zoo. He completed his university education at the Rain Forest Academy, Manaus. He is currently teaching as a Fellow of the Federal University for the Advancement of Furry Animals, Cuiabá. He is a Corresponding Member of Duckpond Academy, Sempione Park, Milano, Italy.

2

Peter's doctoral thesis "On flexibilty: LATEX , latex and rubber" was published at Manaus. His reputation in the academic world is based on his famous study "The Mandelbrot heritage: towards a fractal topology of formicaries". Some of his works arose from a fruitful cooperation with the well known Brazilian-Italian savant Professore P. van Duck.