Brève introduction à une compilation assistée, grâce à arara

Table des matières

1 Introduction

1.1 Présentation rapide

arara est un utilitaire, intégré aux distributions LaTeX, qui permet d'automatiser des actions (définies par des *règles*) de compilation de fichier TeX.

Il existe d'autres *assistants* similaires, comme `latexmk` ou `rubber`, libre à chacun de se renseigner sur la solution qui semble à plus adapter à ses besoins.

Une *spécificité* de arara est d'utiliser Java comme moteur d'exécution, donc une *machine* Java est nécessaire au bon fonctionnement de cette méthode.

1.2 Pourquoi ?

Ayant parfois (souvent ?) besoin d'utiliser des chaînes de compilations différentes (`shell-escape`, `pdflatex`, `lualatex`, etc) j'avais paramétré différentes chaînes de compilation dans mon éditeur TeXstudio, mais je n'étais que *moyennement* satisfait de cette méthode !
Donc une petite recherche sur des *outils* de compilation m'a conduit à me pencher sur des solutions de type *assistants de compilation*.

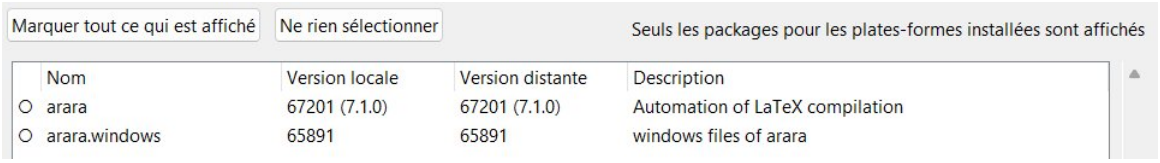
arara a retenu mon attention par :

— sa simplicité de configuration, directement dans le document TeX ;
— sa simplicité à s'adapter à mes différents besoins.

1.3 Installation(s)

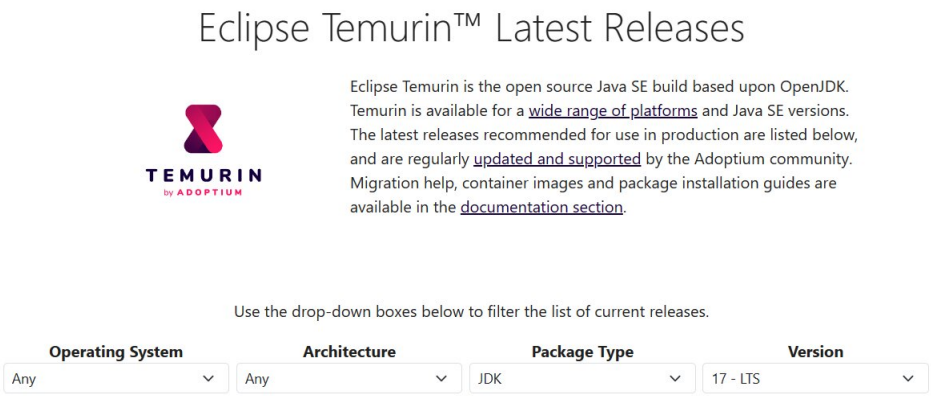
Pour ce qui est du *package* arara, rien de plus simple :

— vérifier dans le gestionnaire de paquets de sa distribution LaTeX qu'il est installé, sinon l'installer grâce à ce même gestionnaire (ligne de commandes ou interface graphique).



Pour ce qui est de l'environnement Java, je conseille l'utilisation de la solution Adoptium (multi-plateformes) qui installera une machine Java prête à l'emploi (avec une inscription des exécutables dans le PATH c'est encore mieux) !

https://adoptium.net/temurin/releases/



1.4 Vérification(s)

On peut vérifier que tout est prêt pour utiliser Java :

— un petit `java --version` dans un terminal permet de vérifier que tout est prêt de ce côté.

Terminal UNiX

```
user@TEST:~$ java --version
openjdk 17.0.8 2023-07-18
OpenJDK Runtime Environment (build 17.0.8+7-Ubuntu-123.04)
OpenJDK 64-Bit Server VM (build 17.0.8+7-Ubuntu-123.04, mixed mode, sharing)
```

Terminal Windows

```
Microsoft Windows [version 10.0.22621.2134]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\user>java --version
openjdk 17.0.3 2022-04-19
OpenJDK Runtime Environment Temurin-17.0.3+7 (build 17.0.3+7)
OpenJDK 64-Bit Server VM Temurin-17.0.3+7 (build 17.0.3+7, mixed mode, sharing)
```

On peut maintenant vérifier que tout est prêt pour utiliser arara :

— un petit `arara --version` dans un terminal permet de vérifier que tout est prêt de ce côté également.

Terminal UNiX

```
user@TEST:~$ arara --version
__ _ _ __ __ _ _ __ __ _
/ _‘ | ’__/ _‘ | ’__/ _‘ |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

arara 7.1.0
Copyright (c) 2023, Island of TeX
arara is released under the New BSD license.

New features in version 7.1.0:
* Add (Lua) project support to arara.
* Use a defined domain-specific file system API instead of ‘java.io.File‘.
* Use header mode by default (‘-w‘ restores the old behavior).

See the full changelog of this release at
https://gitlab.com/islandoftex/arara/-/blob/development/CHANGELOG.md
```

Terminal Windows

```
C:\Users\user>arara --version
__ _ _ __ __ _ _ __ __ _
/ _‘ | ’__/ _‘ | ’__/ _‘ |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

arara 7.1.0
Copyright (c) 2023, Island of TeX
arara is released under the New BSD license.

New features in version 7.1.0:
* Add (Lua) project support to arara.
* Use a defined domain-specific file system API instead of ‘java.io.File‘.
* Use header mode by default (‘-w‘ restores the old behavior).

See the full changelog of this release at
https://gitlab.com/islandoftex/arara/-/blob/development/CHANGELOG.md
```

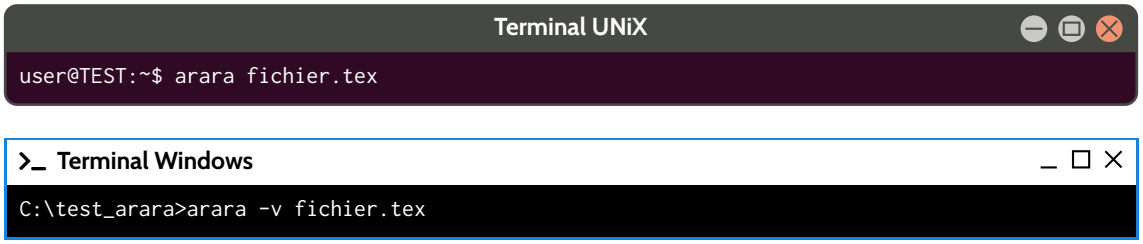
2 Configuration et utilisation

2.1 Idée générale

L'idée générale de arara est de lire, les informations de compilation dans le fichier TEX, grâce à des *commandes* placées au début du fichier, *commandes* qui seront déclarées sous la forme suivante :

```
% arara: commande 1
% arara: commande 2
...

\documentclass{...}
...
```

Il *suffira* ensuite de compiler le fichier TEX à l'aide d'une *simple* ligne de compilation (`-v` pour `-verbose`) :



arara est très *puissant* et paramétrable ! L'idée est ici de montrer des utilisations basiques, mais il est à noter que arara peut être considéré comme un outil de *scirpting* (très) évolué !
La documentation permet de se rendre compte de la richesse de cet outil !

2.2 Première utilisation : travailler sur le moteur de compilation

On peut déjà utiliser arara en précisant le moteur de compilation (et ses éventuelles options), ça sera déjà une bonne base !
Dans ce cas la *règle* se présentera sous la forme suivante :

```
% arara: moteur

\documentclass{...}
...
```

Ainsi, pour compiler un document (très simple) en `pdflatex`, on pourra :

— mettre `% arara: pdflatex` au début du fichier ;
— compiler le fichier grâce à `arara fichier.tex`.

Ainsi, le document `testarara.tex` suivant produira :

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

\begin{document}

Hello world !

\end{document}
```

```
>_ Terminal Windows
C:\test\test_arara>arara testarara.tex
__ _ _ __ __ _ _ __ __ _
/ _‘ | ’__/ _‘ | ’__/ _‘ |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "testarara.tex" (size: 97 B, last modified: 2023-08-11 19:57:13), please wait.

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

Total: 1.44 seconds
```

Le document `testarara.tex` suivant produira :

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

\begin{document}

Hello world !

\end{document}
```

```
>_ Terminal Windows
C:\test\test_arara>arara -v testarara.tex
__ _ _ __ __ _ _ __ __ _
/ _‘ | ’__/ _‘ | ’__/ _‘ |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "testarara.tex" (size: 97 B, last modified: 2023-08-11
20:02:57), please wait.

------------------------------------------------------------------
(LuaLaTeX) LuaLaTeX engine
------------------------------------------------------------------
This is LuaHBTeX, Version 1.17.0 (TeX Live 2023)
restricted system commands enabled.
(./testarara.tex
LaTeX2e <2023-06-01> patch level 1
L3 programming layer <2023-08-03>
(c:/texlive/2023/texmf-dist/tex/latex/base/article.cls
Document Class: article 2023/05/17 v1.4n Standard LaTeX document class
(c:/texlive/2023/texmf-dist/tex/latex/base/size10.clo))
(c:/texlive/2023/texmf-dist/tex/latex/l3backend/l3backend-luatex.def)
(./testarara.aux) (c:/texlive/2023/texmf-dist/tex/latex/base/ts1cmr.fd)
[1{c:/texlive/2023/texmf-var/fonts/map/pdftex/updmap/pdftex.map}]
(./testarara.aux))
406 words of node memory still in use:
3 hlist, 1 vlist, 1 rule, 2 glue, 3 kern, 1 glyph, 4 attribute, 48 glue_spec
, 4 attribute_list, 1 write nodes
avail lists: 2:22,3:4,4:2,5:23,6:2,7:54,9:18
<c:/texlive/2023/texmf-dist/fonts/opentype/public/lm/lmroman10-regular.otf>
Output written on testarara.pdf (1 page, 3356 bytes).
Transcript written on testarara.log.

------------------------------------------------------------ SUCCESS

Total: 2.94 seconds
```

2.3 Deuxième utilisation : travailler sur les options du moteur

On peut également spécifier des *options* au compilateur, grâce à la syntaxe suivante :

```
% arara: moteur: {option1: bool, option2: bool, ...}

\documentclass{...}
...
```

Ainsi on va pouvoir – directement dans le fichier TEX – paramétrer ces options, et ce sans modifier la ligne de compilation !

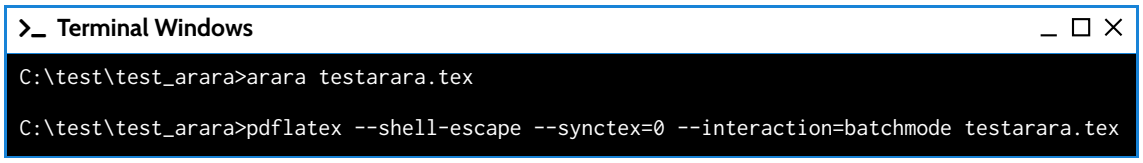
Ainsi, le *règle* suivante :

```
% arara: pdflatex: {shell: yes, synctex: no, interaction: batchmode}
```

demandera de compiler en `pdflatex` :

- avec un accès `shell-escape` ;
- sans la création du fichier `synctex` ;
- avec une *interaction* de type `batchmode`.

Ainsi, les deux lignes de compilation suivantes sont équivalentes :



2.4 Un peu de compilation « conditionnelle »

Une des forces de arara de de pouvoir créer/utiliser les *règles* incluant des tests (la documentation permet de se rendre compte des multiples possibilités !).

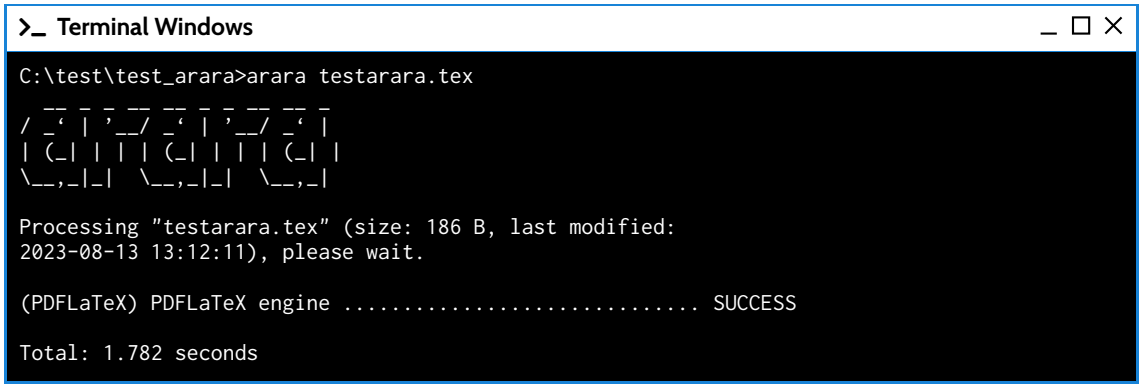
Pour ma part, je me sers de arara pour compiler plusieurs fois si nécessaire (références croisées, sommaire, etc), et dans ce cas je paramètre arara comme suit :

```
% arara: pdflatex
% arara: pdflatex if found('log', '(undefined references|Please rerun|Rerun to get)')

\documentclass{article}

\begin{document}

Hello world !

\end{document}
```

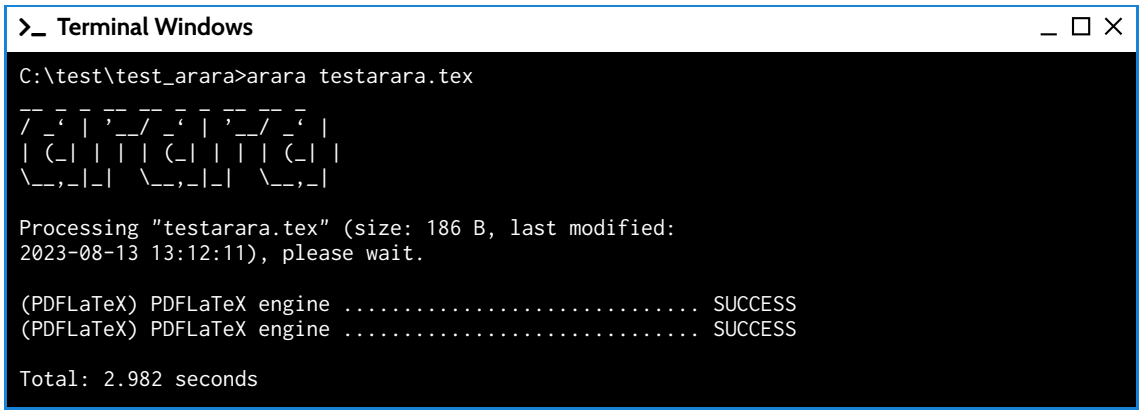
Autrement dit :

- la première *règle* va compiler (sans condition) en `pdflatex` ;
- la deuxième règle va compiler en `pdflatex` uniquement si `undefined references`, `Please rerun` ou `Rerun to get` est trouvé dans le fichier `log`.

En compilant via arara, on peut se rendre compte que la deuxième règle n'a pas été exécutée !



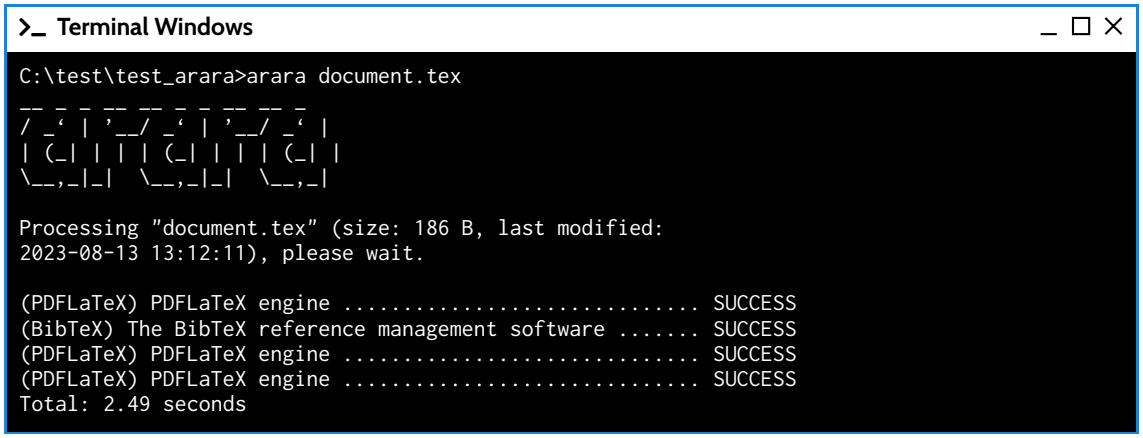
Dans le cas où une deuxième compilation est nécessaire, la sortie sera :



Dans le cas de l'utilisation d'une bibliographie, on pourra utiliser les *règles* suivantes (à compléter si besoin) :

```
% arara: pdflatex
% arara: bibtex
% arara: pdflatex
% arara: pdflatex

\documentclass{article}

\begin{document}
...
...
```

Et la sortie donnera :



3 Compléments

3.1 Erreurs et fichiers log

Lors de la compilation via arara en ligne de commandes (en mode silencieux), la console n'affichera pas forcément les erreurs de manière classique (en fonction du paramétrage de `interaction` notamment). DOnc attention – en ligne de commandes – à la gestion des erreurs (le fichier `log`) est – quoi qu'il arrive – une bonne source pour analyser des erreurs.

```
>_ Terminal Windows
C:\test\test_arara>arara testarara.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "testarara.tex" (size: 109 B, last modified:
2023-08-14 08:09:30), please wait.

(LuaLaTeX) LuaLaTeX engine ............................. FAILURE
Total: 3.254 seconds
C:\texlive\2023\bin\windows\runscript.tlu:921: command failed with exit code 1:
java.exe -jar c:\texlive\2023\texmf-dist\scripts\arara\arara.jar testarara.tex
```

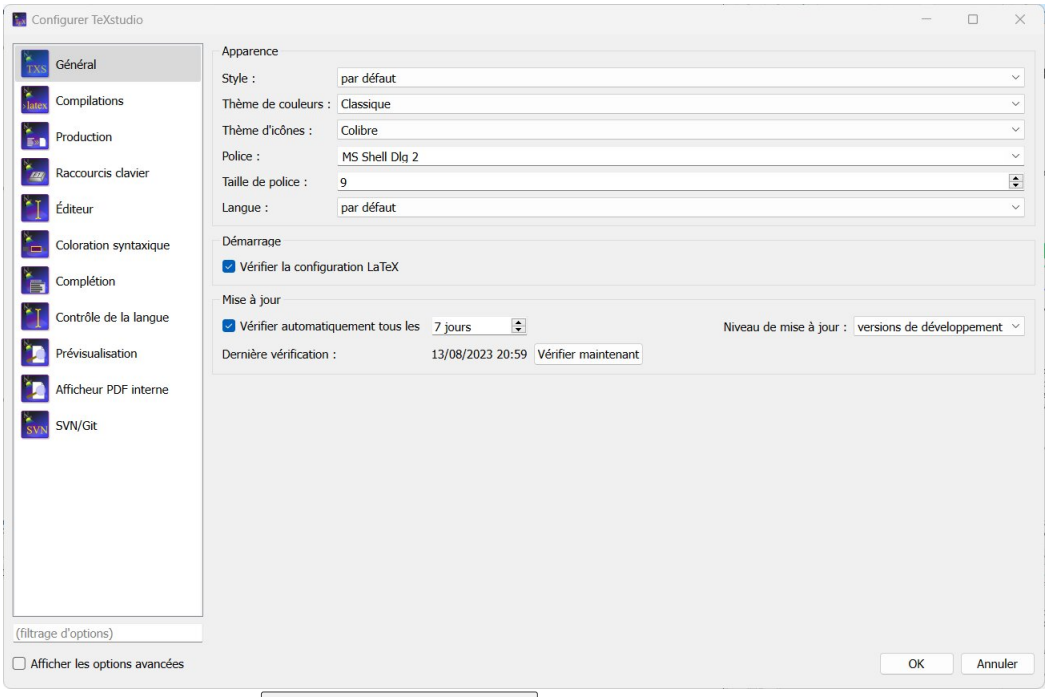
```
>_ Terminal Windows
C:\test\test_arara>arara -v testarara.tex
  __ _ _ __ __ _ _ __ __ _
/ _` | '__/ _` | '__/ _` |
| (_| | | | (_| | | | (_| |
\__,_|_|  \__,_|_|  \__,_|

Processing "testarara.tex" (size: 109 B, last modified:
2023-08-14 08:09:30), please wait.

----------------------------------------------------------------
(LuaLaTeX) LuaLaTeX engine
----------------------------------------------------------------
This is LuaHBTeX, Version 1.17.0 (TeX Live 2023)
restricted system commands enabled.
(./testarara.tex
LaTeX2e <2023-06-01> patch level 1
L3 programming layer <2023-08-11>
(c:/texlive/2023/texmf-dist/tex/latex/base/article.cls
Document Class: article 2023/05/17 v1.4n Standard LaTeX document class
(c:/texlive/2023/texmf-dist/tex/latex/base/size10.clo))
(c:/texlive/2023/texmf-dist/tex/latex/l3backend/l3backend-luatex.def)
(./testarara.aux) (c:/texlive/2023/texmf-dist/tex/latex/base/ts1cmr.fd))
Runaway argument?
{$ \par \end {document}
! File ended while scanning use of \frac .
<inserted text>
\par
<*> testarara.tex

?
```

3.2 Interaction avec un éditeur de fichiers comme TeXstudio

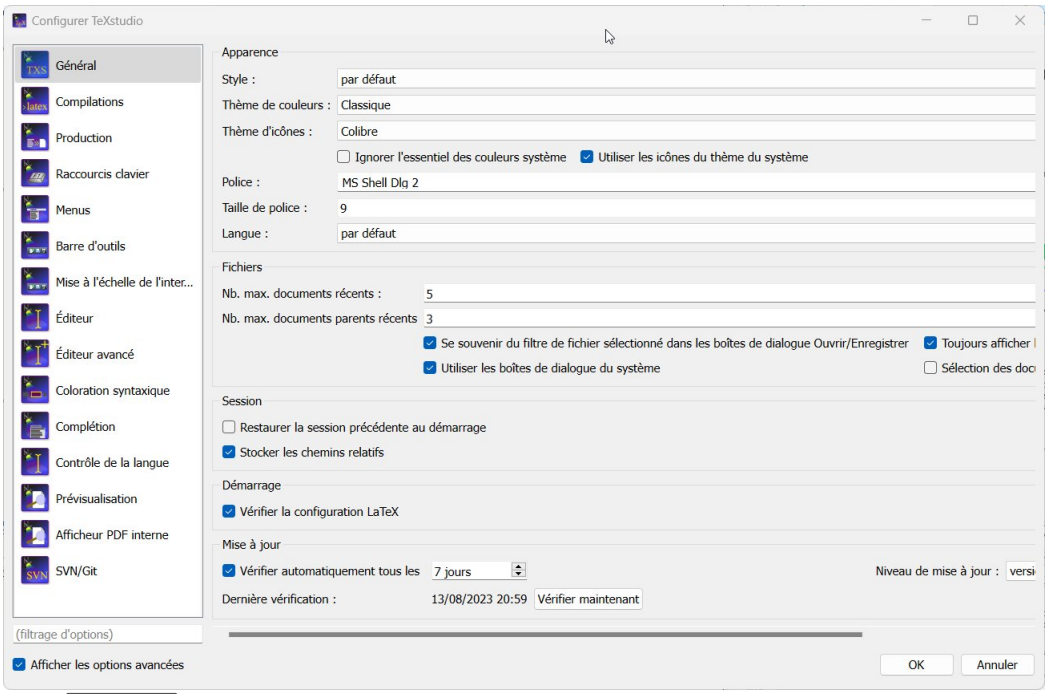
L'éditeur TeXstudio permet de travailler également avec arara, en le déclarant comme compilateur par défaut (et en utilisant éventuellement un *commentaire magique*).

Pour cela, quelques petites manipulations :

— Options > Configuration TeXstudio ;

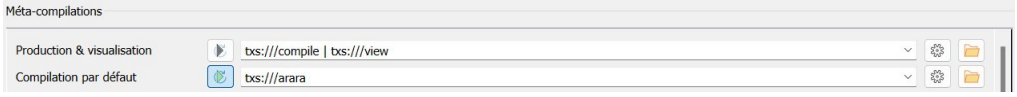


— on coche (si besoin) la case Afficher les options avancées :



— dans l'onglet Production, on créé une nouvelle compilation utilisateur :

— on peut maintenant configurer Compilation par défaut :



Il suffit ensuite de compiler le fichier TEX grâce à F5 (Compilation & Production) !