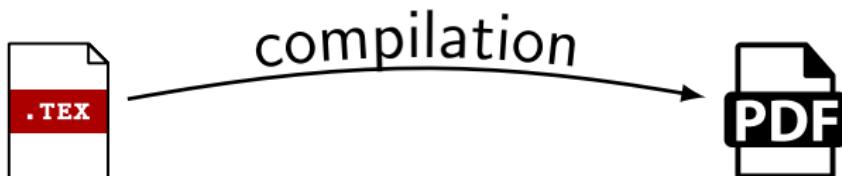


Rédaction mathématique et informatique

Ce que nous avons vu (Rappels) I

- \LaTeX est un **moteur de formatage**



- **orienté contenu** : le style est géré à haut niveau
 - principalement dans la classe (`\documentclass{...}`) choisie
- le squelette d'un document tex est composé essentiellement :
 - d'**un préambule** (début du fichier)
 - et d'**un contenu** (à l'intérieur de l'environnement `document`).

Le code \LaTeX contient

- ▶ **du contenu** (le texte qu'on veut afficher)
- ▶ **des commandes**
 - ▶ commandes simples – e.g., `\LaTeX`, `\tableofcontents`, `\maketitle`
 - ▶ commandes avec argument(s)
 - e.g., `\emph{mis en valeur}` qui donne *mis en valeur*
 - `\section[titre pour la toc]{titre pour le doc}` (c.f. TP1)
 - `\href{mailto:bruno.guillon@uca.fr}{contactez-moi}` qui donne **contactez-moi**
- ▶ les environnements
 - e.g., `\begin{equation*} \dots \end{equation*}`
- ▶ **des caractères spéciaux**
 - ▶ `%` permet de commenter
 - ▶ `$` permet d'entrer dans le mode math ou d'en sortir
 - ▶ `_` et `^` permettent (mode math) d'écrire des indices et des exposants
 - ▶ `\dots`

Ce que nous avons vu (Rappels) III

des commandes usuelles : `\section..., \usepackage..., \begin{equation}... \end{equation}`

le mode math : `a^{2}_{i}`

le mécanisme de référence : (interne : `\label, \ref` ;
et externe : `\url, \href`)

des paquets utiles : `babel, hyperref, mathtools, amssymb`

des bonnes pratiques : importer `inputenc` avec l'option `utf8`
passer des options importantes (e.g., `french`)
à la classe plutôt qu'aux paquets

1. les environnements de liste
2. alignement de texte
3. la gestion des espaces par \LaTeX
4. retour sur le mécanisme de référencement
 $(\backslash\text{label}\{\text{clé}\} \text{ et } \backslash\text{ref}\{\text{clé}\})$
5. approfondissements sur la compilation \LaTeX :
les fichiers auxiliaires

les environnements de liste

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

```
\begin{enumerate}
\item enumerate yo
\item itemize yep
\item description yeah
\end{enumerate}
```

1. enumerate yo
2. itemize yep
3. description yeah

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : `environnement itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

```
\begin{itemize}
\item enumerate yo
\item itemize yep
\item description yeah
\end{itemize}
```

- ▶ `enumerate` yo
- ▶ `itemize` yep
- ▶ `description` yeah

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

<code>\begin{description}</code> <code>\item enumerate yo</code> <code>\item itemize yep</code> <code>\item description yeah</code> <code>\end{description}</code>	enumerate yo itemize yep description yeah
--	---

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

```
\begin{description}
\item enumerate yo
\item itemize yep
\item description yeah
\end{description}
```

enumerate yo
itemize yep
description yeah

Hein ? Ça fait quoi `description` ?

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

```
\begin{description}
\item[enumerate:] yo
\item[itemize:] yep
\item[description:] yeah
\end{description}
```

enumerate : yo
itemize : yep
description : yeah

Hein ? Ça fait quoi `description` ?

→ En fait, `\item` prend un argument optionnel

Trois types :

les listes numérotées : environnement `enumerate`

les listes non-numérotées : environnement `itemize`

les listes descriptives : environnement `description`

À l'intérieur de ces environnements,

la commande `\item` indique les éléments.

```
\begin{description}
\item[enumerate:] yo
\item[itemize:] yep
\item[description:] yeah
\end{description}
```

enumerate : yo
itemize : yep
description : yeah

Hein ? Ça fait quoi `description` ?

→ En fait, `\item` prend un argument optionnel : le titre de l'élément.

(c'est ce que j'ai utilisé en haut !)

Aller plus loin sur les listes

Le paquet `enumitem` permet de gérer au mieux les listes :

Le paquet `enumitem` permet de gérer au mieux les listes :

1. gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs. . .)

Le paquet `enumitem` permet de gérer au mieux les listes :

- 1.** gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

- 2.** gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

Le paquet `enumitem` permet de gérer au mieux les listes :

- 1.** gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

- 2.** gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

- 3.** gérer les références

(“je veux que ma référence affiche *Item 2.1.*”...)

Le paquet `enumitem` permet de gérer au mieux les listes :

1. gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

2. gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

3. gérer les références

(“je veux que ma référence affiche *Item 2.1.*”...)

4. définir d'autres types de liste

(e.g., liste de question q1, q2, ...)

Le paquet `enumitem` permet de gérer au mieux les listes :

1. gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

2. gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

3. gérer les références

("je veux que ma référence affiche *Item 2.1.*"...)

4. définir d'autres types de liste

(e.g., liste de question q1, q2, ...)

5. faire des listes *inline*...

Le paquet `enumitem` permet de gérer au mieux les listes :

1. gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

2. gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

3. gérer les références

(“je veux que ma référence affiche *Item 2.1.*”...)

4. définir d'autres types de liste

(e.g., liste de question q1, q2, ...)

5. faire des listes *inline*...

et bien d'autres choses

Le paquet `enumitem` permet de gérer au mieux les listes :

1. gérer les styles

(chiffres arabes/romain, gras, suivi d'un point, couleurs...)

2. gérer les espaces

(indentation, marges gauche, droite, haute, basse, entre éléments...)

3. gérer les références

("je veux que ma référence affiche *Item 2.1.*"...)

4. définir d'autres types de liste

(e.g., liste de question q1, q2, ...)

5. faire des listes *inline*...

et bien d'autres choses

6. comme reprendre une liste là où on en était.

alignement de texte

Par défaut, le texte est **justifié** (*i.e.*, les lignes apparaissent avec la même longueur). Pour obtenir cela, L^AT_EX utilise deux mécanismes :

1. la césure

i.e., briser un mot en fin de ligne

2. et la variation de taille d'espace

les espaces (entre les mots) peuvent être différents entre deux lignes

Alignement du texte

Par défaut, le texte est **justifié** (*i.e.*, les lignes apparaissent avec la même longueur). Pour obtenir cela, L^AT_EX utilise deux mécanismes :

1. **la césure**

i.e., briser un mot en fin de ligne

La césure nécessite la connaissance de la langue. (merci **babel** !)

2. et **la variation de taille d'espace**

les espaces (entre les mots) peuvent être différents entre deux lignes

Alignement du texte

Par défaut, le texte est **justifié** (*i.e.*, les lignes apparaissent avec la même longueur). Pour obtenir cela, L^AT_EX utilise deux mécanismes :

1. la césure

i.e., briser un mot en fin de ligne

La césure nécessite la connaissance de la langue. (merci **babel** !)

2. et la variation de taille d'espace

les espaces (entre les mots) peuvent être différents entre deux lignes

Alignement du texte

Par défaut, le texte est **justifié** (*i.e.*, les lignes apparaissent avec la même longueur). Pour obtenir cela, L^AT_EX utilise deux mécanismes :

1. la césure

i.e., briser un mot en fin de ligne

La césure nécessite la connaissance de la langue. (merci **babel** !)

2. et la variation de taille d'espace

les espaces (entre les mots) peuvent être différents entre deux lignes

```
bla bla  
\mbox{blablablabla}  
bla.
```

donne

```
bla     bla  
blablablabla  
bla.
```

- * \mbox permet de rendre un mot insécable
(sinon L^AT_EX aurait préféré une césure à ce trop grand espace).

Aller plus loin sur les alignements

Le paquet `microtype` améliore ce mécanisme, notamment en gérant particulièrement les signes de ponctuations en fin de ligne, et en autorisant la variation de largeur de caractère pas seulement les espaces.

il suffit d'importer le paquet¹

1. l'option `babel` est la bienvenue

Aller plus loin sur les alignements

Le paquet `microtype` améliore ce mécanisme, notamment en gérant particulièrement les signes de ponctuations en fin de ligne, et en autorisant la variation de largeur de caractère pas seulement les espaces.

il suffit d'importer le paquet¹

On peut localement changer l'alignement avec les environnements :

`flushleft`

`center`

`flushright`

1. l'option `babel` est la bienvenue

Aller plus loin sur les alignements

Le paquet `microtype` améliore ce mécanisme, notamment en gérant particulièrement les signes de ponctuations en fin de ligne, et en autorisant la variation de largeur de caractère pas seulement les espaces.

il suffit d'importer le paquet¹

On peut localement changer l'alignement avec les environnements :

`flushleft`

`center`

`flushright`

ou avec les commandes :

`\raggedright`²

`\centering`

`\raggedleft`²

1. l'option `babel` est la bienvenue

2. *ragged* signifie accidenté, en lambeau... autrement dit, pas bien aligné.

Aller plus loin sur les alignements

Le paquet `microtype` améliore ce mécanisme, notamment en gérant particulièrement les signes de ponctuations en fin de ligne, et en autorisant la variation de largeur de caractère pas seulement les espaces.

il suffit d'importer le paquet¹

On peut localement changer l'alignement avec les environnements :

`flushleft` `center` `flushright`

ou avec les commandes :

`\raggedright`² `\centering` `\raggedleft`²

Le paquet `ragged2e`² définit des versions améliorées de ces environnements & commandes, ainsi que `justify` et `\justifying`.

1. l'option `babel` est la bienvenue

2. *ragged* signifie accidenté, en lambeau... autrement dit, pas bien aligné.

gestion des espaces

Les espaces et retours à la ligne présents dans le fichier source (`.tex`)
sont interprétés par `LATEX`,
et ne se retrouvent pas nécessairement dans le fichier résultat (`.pdf`).

Source et résultat

- ▶ caractère espace
- ▶ caractère de fin de ligne
- ▶ commandes \LaTeX d'espace
- ▶ autres caractères

Les **espaces et retours à la ligne** présents dans le fichier source (`.tex`) sont interprétés par \LaTeX , et ne se retrouvent pas nécessairement dans le fichier résultat (`.pdf`).

Source et résultat

- ▶ caractère espace
- ▶ caractère de fin de ligne
- ▶ commandes \LaTeX d'espace
- ▶ autres caractères

Les **espaces et retours à la ligne** présents dans le fichier source (`.tex`) sont interprétés par \LaTeX , et ne se retrouvent pas nécessairement dans le fichier résultat (`.pdf`).

espaces côté source \neq espaces côté résultat

- le caractère ‘’ (espace, rendu visible ici) sert à séparer les mots
2 (ou plus) espaces \equiv 1 espace
- un caractère de fin de ligne vaut espace

mais on peut le commenter, e.g.,

```
bla  
bla%  
bla
```

donne

```
bla blabla
```

- le caractère ‘`~`’ (tilde) produit un espace *insécable*
il n'est pas possible de retourner à la ligne sur cet espace

e.g., `Section~\ref{sec/LaTeX}`.

Les retours à la ligne

Il y a deux raisons d'avoir un retour à la ligne dans le pdf :

- ▶ parce qu'on n'a plus de place sur la ligne
- ▶ parce qu'on veut commencer un nouveau paragraphe

Les retours à la ligne

Il y a deux raisons d'avoir un retour à la ligne dans le pdf :

- ▶ parce qu'on n'a plus de place sur la ligne
- ▶ parce qu'on veut commencer un nouveau paragraphe

La première raison ne relève que de la forme

⇒ on laisse \LaTeX gérer cela (merci `babel` et `microtype`!).

Il y a deux raisons d'avoir un retour à la ligne dans le pdf :

- ▶ parce qu'on n'a plus de place sur la ligne
- ▶ parce qu'on veut commencer un nouveau paragraphe

La première raison ne relève que de la forme

 ⇒ on laisse \LaTeX gérer cela (merci `babel` et `microtype`!).

La seconde raison ne relève que du contenu

(un paragraphe est une unité dans le discours)

 ⇒ on indique à \LaTeX les paragraphes,
 en laissant une ligne vide ou via `\par`.



! pas bien, la plupart du temps.



! pas bien, la plupart du temps.

- ▶ `\\" ou \newline` démarre une nouvelle ligne
- ▶ `\linebreak` indique une fin de ligne



! pas bien, la plupart du temps.

- ▶ `\\" ou \newline` démarre une nouvelle ligne
- ▶ `\linebreak` indique une fin de ligne

```
aaa aaa\\
bbb bbb\nnewline
ccc ccc\linebreak
ddd.
```

donne

```
aaa aaa
bbb bbb
ccc      ccc
ddd.
```



! pas bien, la plupart du temps.

- ▶ `\\" ou \newline` démarre une nouvelle ligne
- ▶ `\linebreak` indique une fin de ligne
possibilité de donner un poids (argument optionnel) :
“à quel point encourage-t-on la fin de ligne ici?”

```
aaa aaa\\
bbb bbb\nnewline
ccc ccc\linebreak
ddd.
```

donne

aaa aaa	
bbb bbb	
ccc	ccc
ddd.	



! pas bien, la plupart du temps.

- ▶ `\\" ou \newline` démarre une nouvelle ligne
- ▶ `\linebreak` indique une fin de ligne
possibilité de donner un poids (argument optionnel) :
“à quel point encourage-t-on la fin de ligne ici ?”
- ▶ `\nolinebreak` demande de ne pas terminer une ligne ici
idem, on peut donner un poids.

```
aaa aaa\\
bbb bbb\nnewline
ccc ccc\linebreak
ddd.
```

donne

```
aaa aaa
bbb bbb
ccc      ccc
ddd.
```

Espaces verticaux

- les changements de page gérés automatiquement,

Espaces verticaux

- les changements de page gérés automatiquement,
⚠ mais contrôlables avec `\newpage` et `\pagebreak/\nopagebreak`

Espaces verticaux

- les changements de page gérés automatiquement,
⚠ mais contrôlables avec `\newpage` et `\pagebreak/\nopagebreak`

`\TeX` essaie d'éviter certains résultats indésirables
comme avoir un titre de section en fin de page

- les changements de page gérés automatiquement,
⚠ mais contrôlables avec `\newpage` et `\pagebreak/\nopagebreak`

`LATeX` essaie d'éviter certains résultats indésirables
comme avoir un titre de section en fin de page
pour cela, il fait varier les espaces verticaux
uniformément sur une même page.

- les changements de page gérés automatiquement,
⚠ mais contrôlables avec `\newpage` et `\pagebreak/\nopagebreak`

`LATeX` essaie d'éviter certains résultats indésirables
comme avoir un titre de section en fin de page
pour cela, il fait varier les espaces verticaux
uniformément sur une même page.

On peut tout de même aérer verticalement avec :

skips: des espaces fixes d'une certaine longueur
`(\bigskip, \medskip, \smallskip)`

breaks: des espaces variables en une certaine quantité
qui encouragent le changement de page

`(\bigbreak, \medbreak, \smallbreak)`

- les changements de page gérés automatiquement,
⚠ mais contrôlables avec `\newpage` et `\pagebreak/\nopagebreak`
- `\TeX` essaie d'éviter certains résultats indésirables
comme avoir un titre de section en fin de page
pour cela, il fait varier les espaces verticaux
uniformément sur une même page.

On peut tout de même aérer verticalement avec :

skips: des espaces fixes d'une certaine longueur ⚡
(`\bigskip`, `\medskip`, `\smallskip`)

breaks: des espaces variables en une certaine quantité
qui encouragent le changement de page

(`\bigbreak`, `\medbreak`, `\smallbreak`)

les références

(rappels)

Références internes

but : pouvoir faire référence dans le texte, à un élément ou une partie sans avoir à mettre à jour les numéros en cas de changement.

second but : rendre ces références cliquables grâce à [hyperref](#).

Références internes

but : pouvoir faire référence dans le texte, à un élément ou une partie sans avoir à mettre à jour les numéros en cas de changement.

second but : rendre ces références cliquables grâce à **hyperref**.

principe : 1. on donne une étiquette à l'élément qu'on veut référencer

`\label{elt/a propos de ça}` (ne produit aucun rendu)

2. on fait référence à cette étiquette, là où on veut

`\ref{elt/a propos de ça}` (produit la référence)

Références internes

but : pouvoir faire référence dans le texte, à un élément ou une partie sans avoir à mettre à jour les numéros en cas de changement.

second but : rendre ces références cliquables grâce à `hyperref`.

principe : 1. on donne une étiquette à l'élément qu'on veut référencer

`\label{elt/a propos de ça}` (ne produit aucun rendu)

2. on fait référence à cette étiquette, là où on veut

`\ref{elt/a propos de ça}` (produit la référence)

```
\section{LaTeX}
\label{sec/latex}
Voir Section~\ref{sec/git}.
\section{Git}
\label{sec/git}
Comme vu en
Section~\ref{sec/latex}...
```

1 LaTeX

Voir Section 2.

2 Git

Comme vu en Section 1...

Références internes

but : pouvoir faire référence dans le texte, à un élément ou une partie sans avoir à mettre à jour les numéros en cas de changement.

second but : rendre ces références cliquables grâce à `hyperref`.

principe : 1. on donne une étiquette à l'élément qu'on veut référencer

`\label{elt/a propos de ça}` (ne produit aucun rendu)

2. on fait référence à cette étiquette, là où on veut

`\ref{elt/a propos de ça}` (produit la référence)

```
\section{LaTeX}
\label{sec/latex}
Voir Section~\ref{sec/git}.
```

```
\section{Transition}
De Section~\ref{sec/latex}
à Section~\ref{sec/git}...
```

```
\section{Git}
\label{sec/git}
Comme vu en
Section~\ref{sec/latex}...
```

1 LaTeX

Voir Section 3.

2 Transition

De Section 1 à Section 3...

3 Git

Comme vu en Section 1...

Que peut-on référencer ?

- ▶ les *parties* du documents (sections, sous-sections,...)
- ▶ les *blocs sémantiques* (exemples, théorèmes, équations,...)
- ▶ les blocs flottants (figures, tables,...)
- ▶ les éléments de liste numérotées
- ▶ les notes de bas de page

Que peut-on référencer ?

- ▶ les *parties* du documents (sections, sous-sections,...)
- ▶ les *blocs sémantiques* (exemples, théorèmes, équations,...)
- ▶ les blocs flottants (figures, tables,...)
- ▶ les éléments de liste numérotées
- ▶ les notes de bas de page

il suffit de placer `\label` au bon endroit

Que peut-on référencer ?

- ▶ les *parties* du documents (sections, sous-sections,...)
- ▶ les *blocs sémantiques* (exemples, théorèmes, équations,...)
- ▶ les blocs flottants (figures, tables,...)
- ▶ les éléments de liste numérotées
- ▶ les notes de bas de page

il suffit de placer `\label` au bon endroit
et surtout de donner des **étiquettes uniques !**

Que peut-on référencer ?

- ▶ les *parties* du documents (sections, sous-sections,...)
- ▶ les *blocs sémantiques* (exemples, théorèmes, équations,...)
- ▶ les blocs flottants (figures, tables,...)
- ▶ les éléments de liste numérotées
- ▶ les notes de bas de page

il suffit de placer `\label` au bon endroit
et surtout de donner des **étiquettes uniques !**

Bonnes pratiques:

- ✓ donner des étiquettes sémantiques (de quoi parle l'élément ?) & brèves
- ✓ préfixer les étiquettes avec un code de type d'élément
 - (e.g., '`sec/`', '`ex/`', '`thm/`', '`eq/`', '`fig/`', '`tab/`')
- ✓ utiliser `\eqref` plutôt que `\ref` pour les références aux équations
- ✗ ne pas indiquer de numéro dans l'étiquette ('`sec/Section 3`')

Références cliquables

Si on importe le paquet **hyperref** les références deviennent cliquables.

Si on importe le paquet **hyperref** les références deviennent cliquables.

- ▶ on peut utiliser `\ref*{elt/label}` pour éviter cela.
- ▶ on peut utiliser `\hyperref[elt/label]{cliquez ici}`
pour faire une référence interne avec un texte différent.

Si on importe le paquet **hyperref** les références deviennent cliquables.

- ▶ on peut utiliser `\ref*{elt/label}` pour éviter cela.
- ▶ on peut utiliser `\hyperref[elt/label]{cliquez ici}`
pour faire une référence interne avec un texte différent.
- ▶ on peut utiliser `\phantomsection`
pour définir une partie fantôme (*i.e.*, définir une cible d'hyperliens).

Références externes

... mais `hyperref` permet surtout de faire des références externes :

... mais `hyperref` permet surtout de faire des références externes :

- ▶ `\url{adresse web}` → affiche l'adresse web
et envoie sur cette adresse lorsqu'on clique dessus

- ▶ `\href{adresse web}{texte choisi}` → affiche le texte choisi
et envoie sur l'adresse web donnée lorsqu'on clique dessus

... mais `hyperref` permet surtout de faire des références externes :

- ▶ `\url{adresse web}` → affiche l'adresse web
et envoie sur cette adresse lorsqu'on clique dessus
- ▶ `\href{adresse web}{texte choisi}` → affiche le texte choisi
et envoie sur l'adresse web donnée lorsqu'on clique dessus

voir [TP1.exo3.q17-18].

Aller plus loin sur les références internes

Le paquet `cleveref` permet le référencement intelligent :

- ▶ on écrit `\cref{sec/LaTeX}`
au lieu de `Section~\ref{sec/LaTeX}`

`cleveref` repère tout seul qu'il s'agit d'une section.
cela nécessite l'accès à la langue du document.

Aller plus loin sur les références internes

Le paquet `cleveref` permet le référencement intelligent :

- ▶ on écrit `\cref{sec/LaTeX}`
au lieu de `Section~\ref{sec/LaTeX}`

`cleveref` repère tout seul qu'il s'agit d'une section.
cela nécessite l'accès à la langue du document.

- ▶ on écrit `\cref{sec/LaTeX, sec/git}`
au lieu de `Section~\ref{sec/LaTeX}` et `\ref{sec/git}`

`cleveref` fabrique tout seul la liste de références

Aller plus loin sur les références internes

Le paquet `cleveref` permet le référencement intelligent :

- ▶ on écrit `\cref{sec/LaTeX}`
au lieu de `Section~\ref{sec/LaTeX}`

`cleveref` repère tout seul qu'il s'agit d'une section.
cela nécessite l'accès à la langue du document.
- ▶ on écrit `\cref{sec/LaTeX, sec/git}`
au lieu de `Section~\ref{sec/LaTeX}` et `\ref{sec/git}`

`cleveref` fabrique tout seul la liste de références
- ▶ toute la référence devient cliquable (avec `hyperref`)
 - pas seulement le numéro

Aller plus loin sur les références internes

Le paquet `cleveref` permet le référencement intelligent :

- ▶ on écrit `\cref{sec/LaTeX}`
au lieu de `Section~\ref{sec/LaTeX}`

`cleveref` repère tout seul qu'il s'agit d'une section.
cela nécessite l'accès à la langue du document.
- ▶ on écrit `\cref{sec/LaTeX, sec/git}`
au lieu de `Section~\ref{sec/LaTeX}` et `\ref{sec/git}`

`cleveref` fabrique tout seul la liste de références
- ▶ toute la référence devient cliquable (avec `hyperref`)
– pas seulement le numéro
- ▶ on peut définir de nouveaux éléments référençables.

Aller plus loin sur les références internes

Le paquet `cleveref` permet le référencement intelligent :

- ▶ on écrit `\cref{sec/LaTeX}`
au lieu de `Section~\ref{sec/LaTeX}`

`cleveref` repère tout seul qu'il s'agit d'une section.
cela nécessite l'accès à la langue du document.
- ▶ on écrit `\cref{sec/LaTeX, sec/git}`
au lieu de `Section~\ref{sec/LaTeX}` et `\ref{sec/git}`

`cleveref` fabrique tout seul la liste de références
- ▶ toute la référence devient cliquable (avec `hyperref`)
 - pas seulement le numéro
- ▶ on peut définir de nouveaux éléments référençables.
- ▶ on peut référencer des intervalles avec `\crefrange`.

fichiers auxiliaires de compilation

Comment le moteur fonctionne ?

Lorsqu'on compile, le programme lit le fichier source séquentiellement
⇒ le traitement doit se faire au fur et à mesure.

Comment le moteur fonctionne ?

Lorsqu'on compile, le programme lit le fichier source séquentiellement
 ⇒ le traitement doit se faire au fur et à mesure.

Problème : certaines informations arrivent trop tard, par exemple :

- ▶ les emplacements des étiquettes (`\label`)
- ▶ la table des matières (au début) dépend des titres de sections (après)
- ▶ l'équilibrage des espaces verticaux nécessite de connaître ce qui suit

Comment le moteur fonctionne ?

Lorsqu'on compile, le programme lit le fichier source séquentiellement
 ⇒ le traitement doit se faire au fur et à mesure.

Problème : certaines informations arrivent trop tard, par exemple :

- ▶ les emplacements des étiquettes (`\label`)
- ▶ la table des matières (au début) dépend des titres de sections (après)
- ▶ l'équilibrage des espaces verticaux nécessite de connaître ce qui suit

Solution : stocker des données de travail & compiler plusieurs fois

Comment le moteur fonctionne ?

Lorsqu'on compile, le programme lit le fichier source séquentiellement
 ⇒ le traitement doit se faire au fur et à mesure.

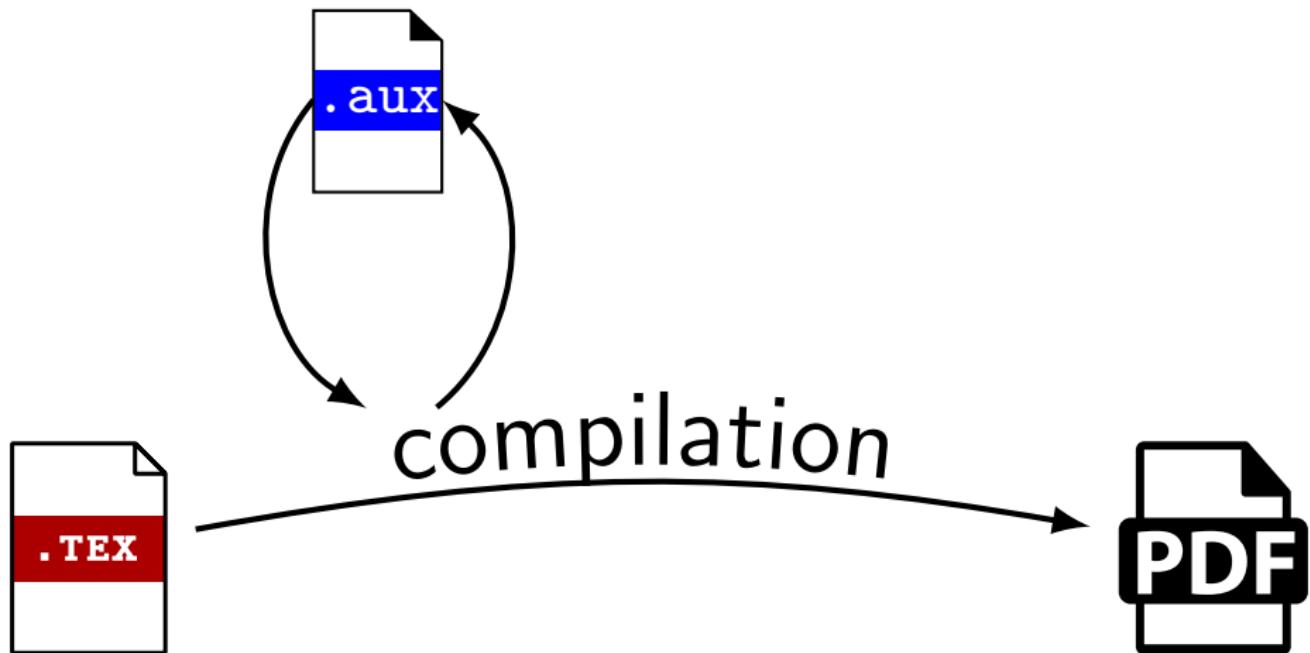
Problème : certaines informations arrivent trop tard, par exemple :

- ▶ les emplacements des étiquettes (`\label`)
- ▶ la table des matières (au début) dépend des titres de sections (après)
- ▶ l'équilibrage des espaces verticaux nécessite de connaître ce qui suit

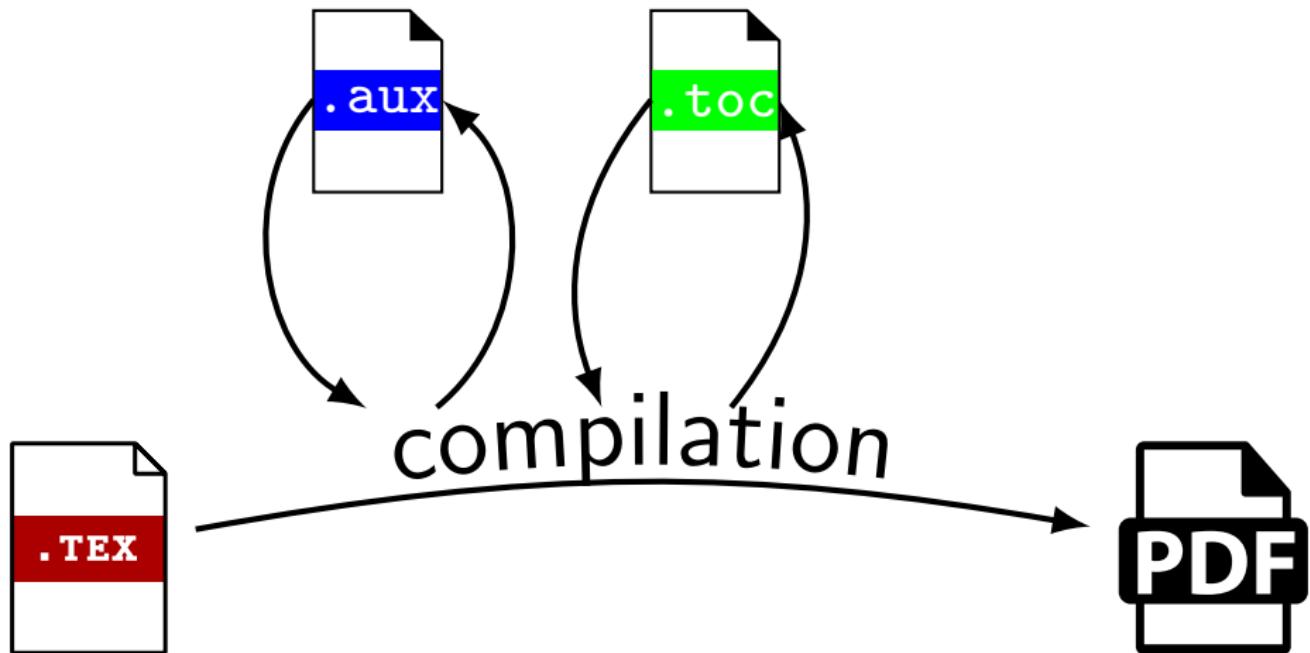
Solution : stocker des données de travail & compiler plusieurs fois



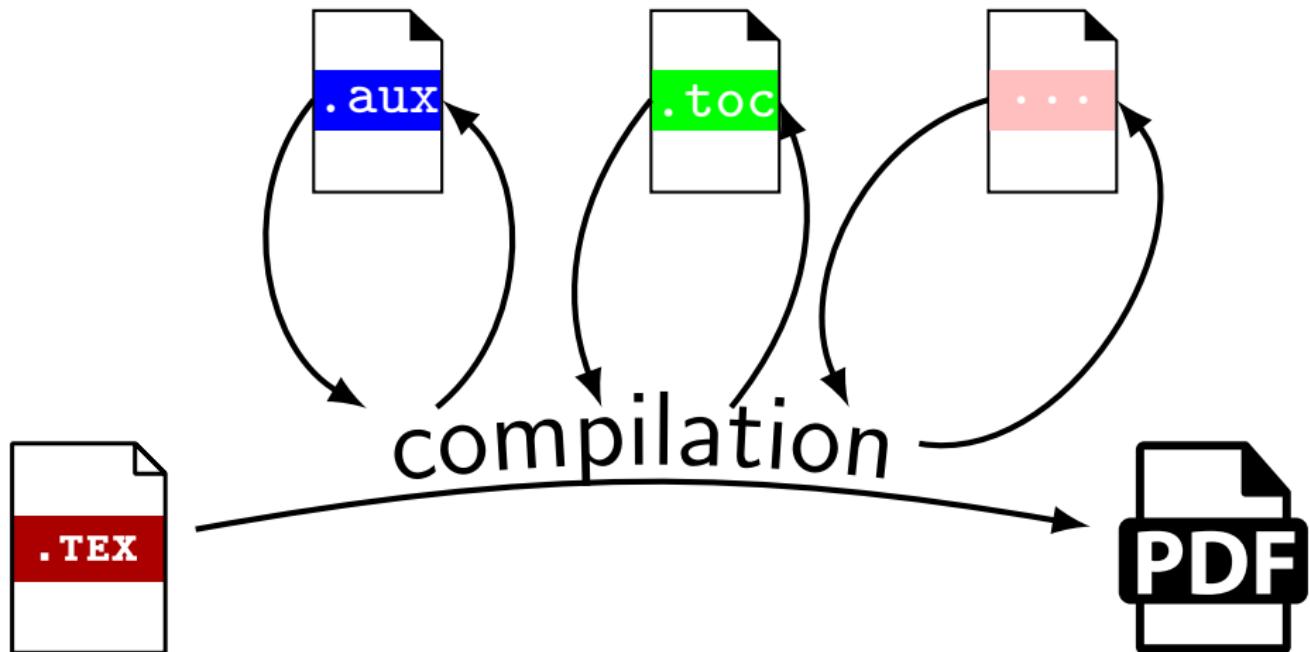
Compilation avec fichiers auxiliaires



Compilation avec fichiers auxiliaires

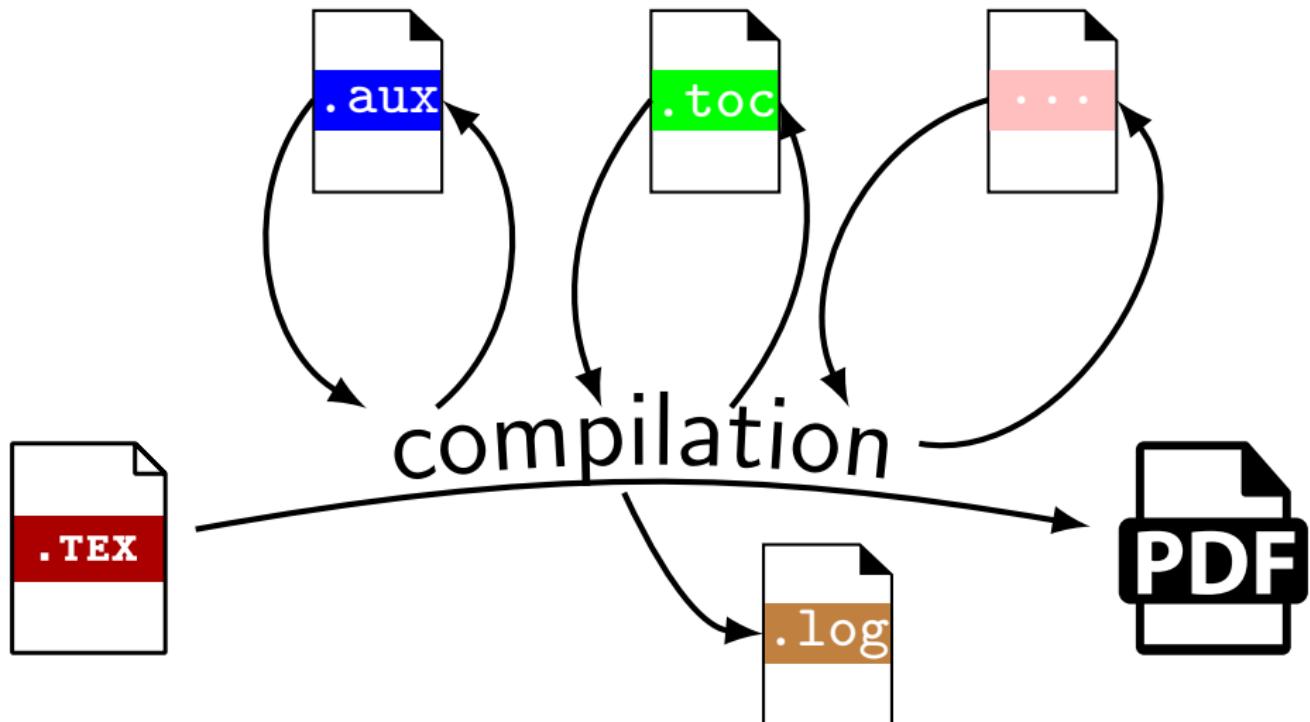


Compilation avec fichiers auxiliaires



Certains paquets créent leur propres fichiers auxiliaires.

Compilation avec fichiers auxiliaires



Certains paquets créent leur propres fichiers auxiliaires.
Le compilateur nous raconte tout ce qu'il fait.

Première compilation :

- ▶ le programme voit les `\ref`, mais ne connaît pas encore l'élément à référencer
 - ⇒ il écrit “??” à la place des références (accompagné d'un *warning*)
- ▶ le programme voit les `\label`,
 - ⇒ il écrit dans le fichier `.aux` l'emplacement des éléments étiquetés (numéro de page, numéro de référence, *etc.* . . .)

Première compilation :

- ▶ le programme voit les `\ref`, mais ne connaît pas encore l'élément à référencer
 - ⇒ il écrit “??” à la place des références (accompagné d'un *warning*)
- ▶ le programme voit les `\label`,
 - ⇒ il écrit dans le fichier `.aux` l'emplacement des éléments étiquetés (numéro de page, numéro de référence, *etc.*...)

Deuxième compilation :

- ▶ le programme voit les `\ref` et connaît l'élément à référencer grâce au fichier `.aux`
 - ⇒ il crée la référence

Première compilation :

- ▶ le programme voit les `\ref`, mais ne connaît pas encore l'élément à référencer
 - ⇒ il écrit “??” à la place des références (accompagné d'un *warning*)
- ▶ le programme voit les `\label`,
 - ⇒ il écrit dans le fichier `.aux` l'emplacement des éléments étiquetés (numéro de page, numéro de référence, *etc.*...)

Deuxième compilation :

- ▶ le programme voit les `\ref` et connaît l'élément à référencer grâce au fichier `.aux`
 - ⇒ il crée la référence

Suivant ce qu'on veut (bibliographie, index, *etc.*...),
d'avantage de compilations peuvent être nécessaires...

Semaine prochaine

- ▶ TP : manipulation du langage
 - ▶ gestion des retours à la ligne, paragraphes
 - ▶ ajout d'un tableau
 - ▶ mode texte/mode math, encore

Semaine prochaine

- ▶ TP : manipulation du langage
 - ▶ gestion des retours à la ligne, paragraphes
 - ▶ ajout d'un tableau
 - ▶ mode texte/mode math, encore

Défi

- ▶ Combien de paragraphes et de mots donne le code suivant ?

\LaTeX:

bla bla%

bla,

%

bla bla%

bla.



?

Semaine prochaine

- ▶ TP : manipulation du langage
 - ▶ gestion des retours à la ligne, paragraphes
 - ▶ ajout d'un tableau
 - ▶ mode texte/mode math, encore

Défi

- ▶ Combien de paragraphes et de mots donne le code suivant ?

\LaTeX:

bla bla%

bla,

%

bla bla%

bla.



?

Semaine prochaine

- ▶ TP : manipulation du langage
 - ▶ gestion des retours à la ligne, paragraphes
 - ▶ ajout d'un tableau
 - ▶ mode texte/mode math, encore

Défi

- ▶ Combien de paragraphes et de mots donne le code suivant ?

\LaTeX:

bla bla%

bla,

%

bla bla%

bla.



\LaTeX : bla bla

bla, bla blabla.