LaTeX en MPI

Montaigne 2023-2024

27 novembre 2023

Un peu d'histoire

TEX est un système logiciel libre de composition de documents, indépendant du matériel utilisé pour la visualisation ou l'impression. Il fut créé à partir de 1977 par le mathématicien et informaticien *Donald Knuth*, excédé par la piètre qualité de la typographie des logiciels d'édition de l'époque.

TEX

Il est principalement conçu pour l'édition de documents techniques et est largement utilisé par les scientifiques, particulièrement en mathématiques, physique, bio-informatique, astronomie et informatique. Il est également extensible et permet notamment l'édition de documents plus complexes. TEX vient de , début du mot , tékhnê (« art, science », en grec ancien), et se prononce /t x/2 ou /t k/, au choix.

Source : `https://fr.wikipedia.org/wiki/TeX`

LaTeX est un langage et un système de composition de documents. Il s'agit d'une collection de macrocommandes destinées à faciliter l'utilisation du « processeur de texte » TeX. LaTeX permet de rédiger des documents dont la mise en page est réalisée automatiquement en se conformant du mieux possible à des normes typographiques. Une fonctionnalité distinctive de LaTeX est son mode mathématique, qui permet de composer des formules complexes.

LaTeX est particulièrement utilisé dans les domaines techniques et scientifiques pour la production de documents de taille moyenne (tels que des articles) ou importante (thèses ou livres, par exemple). Néanmoins, il peut être employé pour générer des documents de types très variés (lettres ou transparents, par exemple). Enfin, de nombreux sites Internet — dont le texte est typiquement mis en forme par d'autres moyens — emploient un sous-ensemble de LaTeX pour composer notamment leurs formules mathématiques.

Source : `https://fr.wikipedia.org/wiki/LaTeX`

Définition

Quelque chose à lire...

Théorème

Quelque chose à lire...

Définition

Quelque chose à lire...

Théorème

Quelque chose à lire...

Théorème (théorème)

Quelque chose à lire...

```
let dfs g =
  let n = size g in
  let visited = Array.make n false in
  let rec dfs u =
    if not visited.(u) then (
      visited.(u) <- true;
      List.iter dfs (succ g u)
    );;
```

Mathématiques

$$\left(\int_{-\infty}^{+\infty} e^{-t^2}\,dt\right)^2 = \int_0^{2\pi} d\theta \left(\int_0^{+\infty} r^2 e^{-r^2}\,dr\right)$$
$$= \sqrt{\pi}$$

Suite de Fibonacci

$$\begin{cases} F_0 = 0 \quad F_1 = 1 \\ \forall n \in \mathbb{N} \quad F_{n+2} = F_{n+1} + F_n \end{cases}$$

Définition (problème de l'arrêt en OCaml)

Le *problème de l'arrêt* consiste à écrire une fonction OCaml :

```
halt: string -> string -> bool
```

prenant en entrées :

- une chaîne `f` contenant le code source d'un programme OCaml,
- une chaîne `e` représentant des entrées pour le programme donné par `f`,

et qui, pour toutes chaînes `f` et `e`, termine en un temps fini en renvoyant `true` si l'exécution du programme `f` sur les entrées `e` termine en temps fini, et `false` sinon.

Théorème

Il n'existe pas de fonction OCaml résolvant le problème de l'arrêt.

Démonstration.

Ce résultat se démontre par l'absurde, en supposant l'existence d'une fonction `halt: string -> string -> bool` répondant à la spécification du problème de l'arrêt et en s'en servant pour construire un programme dont le comportement est contradictoire. Supposons l'existence d'une telle fonction `halt` et considérons alors la fonction `weird` suivante.

```
let weird f e =
  if halt f e then
    while true do print_string "ok" done;
```

Images