Quelques éléments sur les fichiers et bibliothèques en Maple

Philippe Aubry

1 Fichiers

Maple lit et sauvegarde des fichiers de natures différentes. Il reconnaît par l'extension `.m` les fichiers en format interne Maple (voir le paragraphe 1.2).

1.1 Feuille de travail (interface graphique)

1. on la sauve en format `mws`.
2. sauvegarde en mode texte, éditable. Deux modes texte :
   - *Text* classique,
   - *Maple Text*, où les zones de texte de la feuille de travail deviennent précédées d'un signe #.

   L'ouverture de ces fichiers dans Maple en mode texte classique donne un paragraphe de texte dans la feuille de travail. Lorsqu'on ouvre ces fichiers en mode *Maple Text*, les commandes Maple sont prises en compte en tant que telles. Si le fichier a été enregistré en mode *Text*, on ne retrouve pas les paragraphes de texte alors qu'on les retrouve si le fichier a été enregistré en mode *Maple Text*.
3. exportation au format Latex (vous placerez le fichier *maple2e.sty* disponible dans la distribution de Maple pour pouvoir compiler ces fichiers avec LaTeX). Voir aussi la commande `latex` pour exporter des expressions au format LaTeX.
4. exportation au format `html`.

1.2 Expressions

La sauvegarde de la valeur de variables s'effectue avec l'instruction `save` et la lecture par `read`. Un fichier ainsi créé est de type ASCII et peut-être lu dans un éditeur de texte sauf si on suffixe le nom par `.m` qui permet d'obtenir un fichier en format Maple chargeable rapidement.

L'instruction `save x,y,nomfich` sauvegarde les valeurs des variables `x` et `y` dans le fichier `nomfich`. La commande `read` n'admet qu'un seul argument. Ainsi, `read nomfich` charge dans la session les valeurs des variables stockées dans le fichier `nomfich`.

```
> a:=3333:b:=55555:c:=777777:
> save a,b,fichier;
> restart;
> read fichier;
```

$$a := 3333$$
$$b := 55555$$

Le fichier est repéré à partir du répertoire de travail (duquel a été lancé Maple). Il faut délimiter le nom du fichier par des guillemets s'il contient des signes particuliers tels que « . » ou « / ». Le répertoire de travail se visualise par `currentdir()` et se modifie par `currentdir("nouveauRepertoire")` (qui affiche alors le répertoire de travail précédent).

1.3 Fichier de commandes

C'est un fichier texte composé d'une suite de commandes Maple (commentaires autorisés). Il est chargé dans une feuille de travail par la commande `read fichier ;` Mêmes remarques pour les noms de fichiers qu'à la section précédente. La syntaxe peut être vérifiée avec le programme *mint* (en ligne de commande sous Unix, on peut aussi utiliser le programme *maplemint* pour la correction de procédures).

L'existence d'un fichier de commandes nommé `.mapleinit` dans son répertoire principal permet l'exécution de ces commandes au démarrage de Maple (ce sont généralement des chargements de fichiers).

La lecture d'un fichier de commandes est utilisée pour le lancement en tâche de fond. Ce fichier se terminera alors par la commande `quit ;` Dans ce cas on redirige les entrées-sorties. Par exemple, pour exécuter le fichier de commandes `test` et récupérer les résultats dans le fichier `test.out`, on écrira en ligne de commande sous Unix : `maple < test > test.out &`

1.4 Affichage des résultats dans un fichier

Après l'instruction `writeto(nomfich)`, les résultats des commandes sont imprimées dans le fichier `nomfich` au lieu d'être affichées à l'écran. Le prompt et la commande sont aussi imprimés dans le fichier. Si `nomfich` existait alors il est écrasé. L'instruction `appendto(nomfich)` fonctionne de la même manière mais les sorties sont ajoutées à la fin du fichier `nomfich`.

Pour écrire de nouveau sur le terminal, on indique `writeto(terminal)`.

```
> writeto(toto);
> a:=7;
> writeto(terminal);
> a:=6;
```

$$a := 6$$

2 Bibliothèques, packages

Lorsqu'on dispose de beaucoup de fichiers en format interne Maple (c'est le cas pour toutes les procédures dont on dispose par défaut), on a intérêt à les regrouper en une bibliothèque, composée d'un index (`maple.ind`) et de la bibliothèque (*library* en anglais) elle-même (`maple.lib`). L'utilitaire *march* permet de gérer des bibliothèques. La fonction `readlib(f)` permet de charger la procédure `f` de la bibliothèque.

```
> ##rand() fournit un nombre aléatoire de 12 chiffres
> readlib(rand);
```

$$\mathbf{proc}(r)\ \ldots\ \mathbf{end}$$

On peut passer des noms de fichiers en arguments supplémentaires. La plupart des fonctions de Maple sont en fait définies par **f :='readlib('f')' :** . L'exécution consiste en la lecture du fichier `f.m` dans la bibliothèque Maple (valeur de `libname`). Si une valeur est assignée à `f` durant la lecture, ce résultat est retourné sinon il se produit une erreur. On a ainsi :

```
> titi:=readlib('gcd'):
> titi(12,8);
```

$$4$$

Plus généralement, il y a 4 catégories de fonctions en Maple :

1. les "builtin", qui sont dans le noyau (exemple : la fonction `normal`).
2. les fonctions disponibles directement (exemple : `gcd`), chargées automatiquement par un `readlib`.
3. les fonctions à charger à la main avec `readlib` (exemple : `readlib(showtime)`).
4. les fonctions définies dans des packages, disponibles après une instruction **with(nom_package)** ou directement avec **nom_package[f](arguments)**. En effet, un package `p` correspond à une table `p` définie avec `p[f1] :='readlib('`p/f1`')'` etc..., et **with(p)** réalise les instructions **f1 :=p[f1]** etc...

Le source des fonctions est disponible sauf pour celles de **1**. Faire

```
> interface(verboseproc=2);
```

puis `print(gcd)` pour obtenir le source de la fonction `gcd` par exemple.