Manuel d’utilisation de la commande MyTaches

Aide rapide

Code	Effet
`> task`	Commande (alias) qui permet de lister les tâches, à commencer par celles qu'on doit réaliser actuellement. **Options** `--current` les tâches courantes seulement `--today|auj` les tâches du jour et vraiment du jour `--out` les tâches périmées (note : mais qui sont toujours en cours, qui n'ont pas été détruites ou marquées faites) `--near` pour voir les tâches proches, c'est-à-dire celle qui vont commencer dans la semaine où dans le laps de temps défini dans les configurations `--near=<durée>` pour voir les tâches qui démarreront dans la durée `<durée>` (définie par `5d` ou autre durée) `--futur|future` pour voir les tâches qui démarrent dans moins d'un mois (ou autre durée définie dans la configuration) `--far|loin|lointaine` pour voir les tâches lointaines, donc à plus d'un mois ou autre durée définie dans les configurations `--linked|liees|liées` les tâches liées à d'autres tâches **Options supplémentaires** `--id` pour afficher les identifiants des tâches `--cron` pour notifier les tâches qui démarrent ou qui doivent terminer `--lost|perdue` les tâches seules et sans temps
`> task list`	Affichage de la liste des tâches. De nombreuses options permettent de filtrer les tâches
	Options Les mêmes que pour la commande `> task`
`> task aide`	Affichage de l'aide internet (pas ce fichier)
`> task manuel`	Pour ouvrir le fichier manuel (ce fichier en PDF) **Options** `--edit` Ouvre la fichier Markdown
`> task add|create`	Pour ajouter une tâche
`> task mod|edit[ id=<id>]`	Pour modifier une tâche (la tâche d'identifiant donné s'il est donné)
`> task today`	Affiche les tâches du jour

Création d'une tâche

`> task add` ou `> task create` pour créer une nouvelle tâche.

Définir un temps

Pour définir un temps, les raccourcis sont possibles, comme `dem` pour `demain` ou `mar` pour mardi.

Nature de la tâche en fonction de son temps

Le principe d'une tâche de **MyTaches** est qu'elle est reportée au lendemain lorsqu'elle n'a pas été exécutée à temps (comme dans MonFlux).

Valeurs de la tâche

Pour donner une valeur nulle

Pour donner une valeur nulle (`null`) à une propriété, il faut explicitement entrer la valeur `NULL` à la question demandée.

Entrée d'une date et d'une heure

L'entrée de la date et de l'heure est grandement facilitée par les valeurs par défaut. Par exemple, si on ne met qu'un jour (p.e. `14`), la date est celle du 14 du mois courant, à 0:00. Si on indique seulement une heure, c'est l'heure indiquée du jour courant. Si on n'indique pas l'année, c'est l'année courante, etc.

Tâches parallèles

MyTaches utilise le principe des tâches parallèles qui permet d'avoir plusieurs tâches qui s'exécutent en parallèle avant qu'une autre tâche se déclenche. Ou pour le dire suivant : **la tâche qui suit des tâches parallèles ne se déclenchent que lorsque toutes les tâches parallèles ont été exécutées**.

Pour définir qu'une tâche est parallèle à une autre, on définit sa propriété "Parallèle à...".

Exécution d'un code

On peut obtenir l'exécution d'un code quelconque en définissant la propriété `:exec` c'est-à-dire "code à exécuter" au moment de la création/modification de tâche.

Ce code doit impérativement être du code `shell`. Pour exécuter un code ruby, on peut utiliser :

```
ruby -e "<code ruby à exécuter>"
```

Ou :

```
ruby << CODER
puts "Bonjour tout le monde !"
CODER
```

Noter que si le résultat doit être envoyé quelque part, dans un fichier particulier, il faut uiliser la formule :

```
ruby > ./le/fichier.txt << CODER
puts "Dans le fichier.txt"
CODER
```

> Note : donc mettre le `> path/to/file` avant `<<`.

Choix d'une application

Lorsque l'on choisit une application on bénéficie de plusieurs choses :

- son icône est utilisée pour le message (c'est plus sympa)
- l'application est lancée lorsque l'on clique sur la tâche (et là... ça devient vraiment plus intéressant... — cf. la suite)

Le lancement de l'application ouvre des possibilités infinies. On peut par exemple fabriquer une application juste pour faire une chose en particulier. C'est le cas par exemple pour l'application Icare (`Icare.app`) qui 1) affiche un papillon dans sa notification et 2) ouvre le fichier des commentaires quand on clique sur la notification. Voir l'exemple de l'application Icare.

Pour utiliser une icone spécifique

Par défaut, la notification utilise l'icone de MonFlux...

Avec `terminal-notifier` on ne peut malheureusement pas utiliser `-appIcon` pour définir l'icone de l'applcation. Pour choisir l'icone, il faut utiliser l'option `-sender` qui attend comme argument l'ID de l'application dont il faut prendre l'icone. Par exemple : `-sender com.apple.Safari` pour utiliser l'icone de l'application Safari.

Icone d'une application existante

Un grand nombre d'applications sont déjà enregistrées pour le choix. Mais si c'est l'application voulue n'est pas présente dans la liste, il faut soit l'ajouter à la liste soit donner explicitement son `Bundler Identifiant` :

- trouver l'application dans le dossier `/Applications`,
- demander l'affichage de son paquet,
- lire la propriété `CFBundleIdentifier` dans `infos.plist` (ou "Bundle Identifier" si on ouvre le fichier dans Xcode),
- la copier-coller dans l'invite.

Icone personnalisée

C'est un peu plus compliqué pour utiliser une icone personnalisée. Il va falloir créer une application qui aura cette icone.

Pour ce faire, le plus simple est de suivre la procédure décrire dans l'aide générale sur le cloud, `iCloud Drive > _AIDES_ > Créer une application > Creer_une_app_MAC.md`

Copier-coller l'ID choisi dans l'invite pour choisir une application.

> Note : il faudra autoriser une première l'application à recevoir des notifications. Ça se fera au premier appel.

Les paramètres temporels d'une tâche

On peut définir trois paramètres temporels de la tâche :

- le temps de début de la tâche,
- le temps de fin de la tâche,
- la durée de la tâche,
- rappels (cf. ci-dessous).

Tous ces paramètres sont optionnels.

Rappels de la tâche

Le paramètre de rappel de la tâche permet d'indiquer quand la tâche doit être rappelée.

Rappel : en fonctionnement normal, la tâche n'est indiquée (notifiée) que lorsqu'elle arrive en début d'échéance ou en fin d'échéance. Le rappel permet de notifier plus souvent ce qui doit être fait. C'est surtout pratique pour les tâches longues.

Pour définir le rappel, il suffit de choisir d'éditer cette propriété et de répondre aux questions.

Temps d'une tâche

Le temps peut être fourni de différentes manières :

- de façon complète : `JJ/MM/AAAA H:MM`,
- de façon tout à fait raccourcie : `H:MM` (la date de la tâche sera mise à aujourd'hui),
- de façon partielle : `J/MM` (l'année courante, à 0:00)
- juste avec le jour et l'heure : `JJ H:MM` (le jour du mois courant à l'heure donnée)

Tâche sans aucun temps

Il s'agit donc d'une tâche qui ne définit ni son temps de départ ni son temps de fin ni sa durée.

Elle est ignorée des tâches courantes et ne sera prise en compte que lorsque un de ces éléments sera défini :

- son temps de début (:start)
- son échéance (:end)
- sa tâche précédente (:prev) si elle définit un temps

Elle permet donc de définir une tâche à utiliser plus tard.

Fabrication d'une application

- Créer son dossier dans `/Applications/MyTaches/`,
- lui donner son nom avec l'extension `.app` (pour l'exemple `MonApp.app`),
- créer à l'intérieur un dossier `Contents`
- créer à l'intérieur un fichier `Info.plist`
- dans ce fichier coller le code suivant en remplaçant `MonApp` et `monapp` par les valeurs de l'application :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>MonApp</string>                    <-- ICI
    <key>CFBundleExecutable</key>
    <string>monapp</string>                    <-- ICI
    <key>CFBundleIdentifier</key>
    <string>phil.app.MonApp</string>           <-- ICI
    <key>CFBundleVersion</key>
    <string>1.0.0</string>
    <key>CFBundleGetInfoString</key>
    <string>INFO MonApp</string>               <-- ICI
    <key>CFBundleShortVersionString</key>
    <string>1.0</string>
  </dict>
</plist>
```

- créer un dossier `Contents/MacOS`
- créer un fichier du nom défini dans `CFBundleExecutable` ci-dessus (par exemple `monapp`) (sans extension)
- ouvrir ce fichier dans `Sublime Text` par exemple
- lui donner le type `ruby` si on écrit en ruby
- coder ce qu'il faut faire (sans oublier le BOM)

```
rb[TAB]
# encoding: UTF-8

# ... le code à exécuter quand on clique sur la notification ...
```

- ouvrir un Terminal à ce fichier et le rendre exécutable (p.e. `chmod +x ./monapp`)
- créer un fichier `PNG` pour l'**ICONE DE L'APPLICATION** (en copian par exemple ceux du dossier `/Application/MyTaches/Chantier/`)
- demander les infos de l'application (la sélectionner et faire CMD-i) et glisser-déposer ce fichier PNG à l'endroit de l'icône
- l'icône est aussitôt attribuée.
- dans le fichier `Tache_edition.rb` de MyTaches, ajouter cette applications à la liste.
- utiliser une première fois cette application pour l'ajouter aux préférences,

  > Note : dans les préférences elle peut encore avoir l'icone du Terminal, mais ça sera réglé au cours d'un prochain rechargement/allumage de l'ordinateur)

- régler les notifications voulues (normalement, juste dans le centre de notifications)
- That's it!

> Note : il est possible qu'il faille attendre un rechargement pour que l'icône soit vraiment prise en compte dans les notifications.