Un interpreteur de programme comme java, perl, clojure, go ou nodejs, permet d'exécuter des programme écrits ou compilés dans un langage spécifique. Pour le lancer on utilise la commande `<nom de l'interpréteur> <nom du programme à interpréter> <paramètres>`. Par exemple dans le cadre de java, on lance la commande `java monpackage.Test 1`, pour nodejs on lance la commande `node toto.js`. Dans les deux cas, il faut que le programme correspondant soit contenu dans un fichier disponible sur votre disque dur.

Tous ces programmes s'installent quasiment de la même manière. Je décris ici la démarche sur un linux ou un macos.

- Lancer l'interpréteur dans une fenêtre de commande pour vérifier qu'il est installé.

```
~$ nodejs
```

Si vous obtenez une réponse du style `-bash: node: command not found` c'est que votre programme n'est pas installé. (Il est possible que le message soit écrit en français). Dans ce cas il faut installer votre programme.

- Aller sur le site web du fournisseur de programme. Par exemple en faisant un `google -> node javascript`, puis téléchargez la version binaire de votre programme. Je vous suggère de le télécharger sur un répertoire local de votre machine. En TC par exemple `/tmp`, ce qui évite de surcharger un serveur qui ne sert pas à cela. La commande classique pour un fichier tgz est `tar ztvf toto.tgz` pour lister le contenu et `tar zxvf toto.tgz` pour extraire le contenu. (Mais souvent un click suffit).

```
/tmp $ tar zxvf node-xxx.tgz
node-xxx/lib/
node-xxx/lib/aa.bin
node-xxx/lib/aaabb.txt
node-xxx/bin/node
...
/tmp $
```

- Vérifiez qu'à partir de votre zone d'installation vous accédez au programme. Par exemple en tapant son chemin d'accès complet.

```
~$ cd
~$ /tmp/node-v23-x86/bin/node`
>
```

Notez souvent le changement de `prompt` quand vous passez dans l'interpréteur interactif. Il

est alors souvent utile de trouver comment en sortir `CTRL-D` ou `CTRL-C` font souvent l'affaire.

- Un dernier point consiste à pouvoir invoquer votre commande de n'importe où de votre environnement. Dans l'exemple précédent, si vous lancez la commande `node` sans chemin absolu, vous obtiendrez toujours l'erreur `command not found`. Il est donc nécessaire de l'ajouter à l'environnement de votre interpréteur de commande (tient encore un interpréteur...). Pour cela il faut modifier votre `PATH` qui contient la liste des chemins testés pour trouver les commande dans votre interpréteur shell. Pour modifier votre variable PATH, on place la commande dans votre environnement `.bashrc ou .bash_profile`. Editez un de ces fichiers, et ajoutez une ligne du style

```
export PATH=$PATH:/tmp/node-xxx/bin
```

Qui indique à votre interpréteur shell, de déclarer la variable PATH, comme étant l'ancienne variable PATH, concaténée du répertoire `/tmp/node-xxx/bin`. Mais cette commande n'est pas exécutée dans votre environnement. Si vous êtes dans votre shell et que vous tapez la commande.

```
~$ echo $PATH
```

Votre nouveau répértoire n'est pas pris en compte, sauf si vous avez ouvert une nouvelle fenêtre et que votre interpréteur a bien chargé votre fichier .bashxxx (ce qui n'est pas toujours le cas). Pour charger votre nouvel environnement dans la fenêtre courante, on utilise la commande `source` ou son alias `.`

```
~$ source ~/.bashrc
ou
~$ . ~/.bash_profile
```

est donc une commande de l'interpréteur shell, qui charge le fichier, et qui par exemple la commande `export variable=valeur` maintient la valeur de la variable dans la mémoire de la fenêtre de commande. Vous pouvez maintenant lancer la commande `node` et faire le TD...

Pour savoir si une commande est disponible dans le `PATH` courant de votre fenêtre courante, vous pouvez utiliser la commande which

```
~ $ echo $PATH
/opt/local/bin:/opt/local/sbin:/opt/local/bin:/opt/iballot/node-v0.10.13-
darwin-
x64/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/local/go/bin:/opt/
wiko/adt-bundle-mac-x86_64-20131030/sdk/platform-tools:/opt/wiko/adt-bundle-
mac-x86_64-20131030/sdk/tools
~ $ which node
/opt/iballot/node-v0.10.13-darwin-x64/bin/node
```