TP 1 : Configuration, XEmacs et Java

Informatique Fondamentale (IF1)

Semaine du 24 septembre 2007

L'objectif de ce tp est de commencer à vous faire découvrir le langage *Java* et de faire apparaître les avantages qu'il peut y avoir à utiliser l'éditeur de texte *xemacs* pour programmer en *Java*.

Par ailleurs, la section 1, indépendante des suivantes, a pour but de vous introduire ce que l'on appelle les *fichiers de configuration*.

Premiers pas avec XEmacs

Le système d'exploitation que vous utilisez dans les salles du SCRIPT (*FreeBSD*), met à votre disposition différents éditeurs de texte (*kwrite*, *nedit*, *vi*, *xemacs* etc.). Nous utiliserons ce semestre, *xemacs*, qui est particulièrement bien adapté à la programmation.

Lancez le programme *xemacs* et regardez dans les différents menus, les commandes proposées (n'y passez pas trop de temps non plus, vous les découvrirez au fur et à mesure des séances de tp). Les commandes de base pour manipuler les fichiers sont dans le menu *Fichier* (*Nouveau* pour créer un fichier, *Ouvrir* pour éditer un fichier existant, *Enregistrer* pour écrire la version actuelle du fichier sur le disque dur, etc.). Remarquez la présence, sur la droite de chaque commande, la présence du *raccourci clavier* correspondant (où `C` représente la touche `Ctrl`).

Note : sur certaines distributions LINUX (comme par exemple la JUPPIX que vous pourrez utiliser pour travailler chez vous), *xemacs* n'est pas installé par défaut ; vous pourrez alors utiliser *emacs*, qui est très semblable.

1 Configuration

Les *fichiers de configuration* permettent à l'utilisateur de personnaliser les applications selon ses préférences. Ces fichiers doivent se situer dans le répertoire *home* de l'utilisateur et leurs noms doivent commencer par un point (`.`). Nous allons voir un exemple en modifiant le comportement du *shell*.

1. A l'aide d'xemacs, créez, dans votre répertoire *home*, un fichier appelé `.bashrc` contenant (exactement) les lignes suivantes :

```
PS1='\w$ '
export PS1
```

Vous pouvez procéder de deux manières différentes :
- soit lancer *xemacs* (en tapant la commande `xemacs`) dans un shell, puis utiliser la commande *Ouvrir* pour créer votre fichier,
- soit créer directement votre fichier en lançant *xemacs* avec le nom de votre fichier en argument de la commande (c'est-à-dire : `xemacs .bashrc`).

Dans aucun cas il ne faudra travailler dans le tampon `*scratch*` qui est présent à l'ouverture de xemacs.

2. N'oubliez pas de sauvegarder votre fichier, puis ouvrez une nouvelle fenêtre shell. Que remarquez vous ?

   Créez un nouveau répertoire appelé `rep1`, puis déplacez-vous dans ce dossier. Que remarquez vous ?

   Revenez dans votre répertoire *home*, supprimez le répertoire `rep1` (puis vérifiez que vous l'avez bien supprimé).

3. A l'aide de la commande `ls`, inspectez le contenu de votre répertoire *home*. Que remarquez-vous ? Quelle est la différence si vous tapez `ls -a` ?

Vous verrez par la suite que les fichiers de configuration permettent de spécifier de nombreuses préférences pour chaque application (à l'instar de `.bashrc` pour le `shell`). N'hésitez pas à aller glâner vous-même des informations à ce sujet (sur internet, par exemple), si cela vous intéresse.

2 Découverte de l'interaction d'XEmacs avec Java

Un fichier contenant du code source Java doit avoir un nom qui se termine par "`.java`".

1. Créez un sous-répertoire de *home* que vous nommerez `IF1_tp1` (vous pourrez par la suite ranger ce répertoire où bon vous semble).
2. Lancez xemacs dans ce répertoire.
3. Dans un fichier que vous nommerez `Bonjour.java`, écrivez le programme suivant en appuyant sur la touche *tab* au début de chaque ligne (cela permet de respecter l'*indentation*, c'est-à-dire le nombre d'espaces au début de chaque ligne) :

```
import fr.jussieu.script.Deug;
public class Bonjour {
    public static void main (String[] args) {
        Deug.println("Bonjour.");
    }
}
```

4. Enregistrez votre fichier. Remarquez que la coloration syntaxique et l'indentation vous permettent d'avoir des repères visuels dans votre programme (pour se rendre compte rapidement qu'on a oublié un point-virgule, par exemple).

   Note : Contrairement à *xemacs*, la coloration syntaxique n'est pas toujours activée par défaut dans *emacs*, mais vous pourrez l'activer à l'aide du menu.

3 Premiers pas en Java et méthode Coué

1. Dans le répertoire `IF1_tp1`, créez un fichier `Jaime.java` , et écrivez-y le programme suivant :

```
import fr.jussieu.script.Deug;
public class Jaime {
    public static void main(String[] args) {
        Deug.println("Java, c'est pas de la menthe à l'eau.");
    }
}
```

   Sauvegardez le, et vérifiez à l'aide de la commande `ls` que le fichier a bien été créé (vous pouvez également utiliser les commandes `cat`, `more` et `less` pour visualiser directement son contenu sans passer par un éditeur de texte).

2. Tapez la commande suivante :

```
javac Jaime.java
```

   Pouvez-vous dire quels fichiers ont été créés ?

   La commande `javac` *compile* le *fichier source* `.java` en un fichier `.class`. Ce processus sera étudié en détail en cours. Le fichier `Jaime.class`, dit *fichier bytecode*, contient un code dit *bytecode*, qui peut être exécuté par la commande `java`. Exécutez-le donc en tapant :

```
java Jaime
```

3. Faîtes de même avec le fichier `Bonjour.java`, et admirez le résultat.

4. Enlevez le point-virgule de la ligne 4 du fichier `Jaime.java` et compilez-le à nouveau. Que se passe-t-il et qu'en déduisez-vous ?

5. Toujours dans le répertoire `IF1_tp1`, créez un fichier `Division.java` qui contient le code suivant :

```
import fr.jussieu.script.Deug;
public class Division {
    public static void main(String[] args) {
        int n, r;
        Deug.println("Entrez un entier");
        n = Deug.readInt();
```

```
        r = 2007/n;
        Deug.println("Le resultat est : " + r);
    }
}
```

Ce programme demande à l'utilisateur d'entrer un nombre entier n, puis affiche la partie entière de $2007/n$.

Compilez ce programme, vérifiez que le fichier *bytecode* a été créé, puis testez le programme. Que se passe-t-il si vous entrez 0 ?

4 Le déboggage... ou la joie de programmer

Vous verrez qu'au début, une grande partie de votre temps de programmation sera occupée par ce que l'on appelle le *déboggage*, autrement dit, chercher la petite erreur de frappe qui empêche votre programme de compiler correctement. Rassurez-vous, avec l'habitude, on fait de moins en moins d'erreurs... ou en tout cas, on les repère de plus en plus vite.

A l'aide de la commande `wget`, téléchargez le fichier `Bonjour2.java` situé à l'adresse suivante :

`http://www.liafa.jussieu.fr/~limouzy/enseignement/if1/Bonjour2.java`

Essayez de compiler ce fichier. Que se passe-t-il ?

En vous aidant des messages affichés par le compilateur, effectuez les modifications nécessaires afin que le fichier puisse être compilé sans erreur.

Une fois le programme compilé, exécutez-le et admirez le résultat. Pouvez-vous afficher `Bonjour Java!!` au lieu de `Bonjour Monde!!` (sans changer le programme) ?