Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2019 – 2020
DEV1 – ENVL

DEV1 – ENVL – Laboratoires d'environnement

TD 3 – Java sur linux

Résumé

Lorsqu'on programme en JAVA sur LINUX, on peut, comme sur WINDOWS, utiliser un environnement de développement comme NETBEANS. Mais on peut aussi tout faire en mode console, c'est ce que nous allons voir ici. Nous en profiterons pour apprendre de nouvelles notions liées à Linux.

Table des matières

Dans votre répertoire `home/dev1`, créez un répertoire `td3`. Ce répertoire contiendra tous les fichiers que vous allez créer aujourd'hui.

Version du document
10 mai 2019

1 Quelques notions supplémentaires de Linux

Avant d'attaquer le cœur du TD, à savoir le compilation et l'exécution de programmes Java, voyons quelques éléments e Linux qui vous seront utiles.

1.1 La ligne de commande

Commençons par quelques astuces pour vous faciliter la vie lorsque vous entrez des commandes.

La complétion de commandes

Parfois, vous devez entrer une commande assez longue parce que les noms de fichiers sont longs et/ou nombreux. Linux offre plusieurs facilités pour simplifier l'entrée de longues commandes.

Lorsque vous appuyez sur la touche `TAB`, le shell tente de compléter le début de commande que vous avez déjà tapé. Si plusieurs possibilités existent, elles sont affichées si vous appuyez 2x sur `TAB`.

Tutoriel 1 **La complétion de la commande**

Supposons que vous ne vous rappeliez plus très bien de la commande qui permet de modifier le mot de passe. Vous vous rappelez juste qu'elle commence par `pas`.

- Tapez `pas` puis appuyez 2x sur la touche `TAB`.
- Entrez un `s` puis appuyez à nouveau sur `TAB`.

La touche de tabulation permet également de compléter un nom de fichier.

Exercice 1 **La complétion des noms de fichiers**

1. Dans votre dossier `td3`, copiez le fichier `monfichieraunomtellementlongquilmeparaitpeuprobabledeletaper2xsanserreur` qui se trouve dans le dossier `/eCours/java/td/td3`.
2. Affichez le contenu de ce fichier en évitant de retaper son nom.

Revenir à une commande précédente

Il arrive souvent qu'il faille entrer une commande qu'on a déjà écrite il y a peu (ou en tout cas fort proche de ce qu'on a déjà écrit). C'est là que les flèches viennent à notre secours.

La flèche vers le haut permet de revenir aux commandes précédentes et de les modifier. À utiliser sans modération...

Historique des commandes

- `history` : affiche la liste des dernières commandes entrées. Chaque commande est numérotée.
- `!numéro` : ré-execute la commande de numéro donné.
- `!cmd` : ré-execute la dernière commande qui commençait par `cmd`.

1.2 Les fichiers cachés

Les fichiers cachés

Un **fichier caché** est un fichier dont le nom commence par « . » (un point).
Idem pour un dossier.

Par défaut, les fichiers cachés ne sont pas montrés par la commande `ls`. Pour les voir, utiliser l'option `a` : `ls -a`. C'est surtout utilisé pour des fichiers de configuration.

Exercice 2

Les fichiers cachés de la home

Regardez s'il existe des fichiers cachés dans votre répertoire personnel.

2 L'éditeur nano

Un éditeur de texte, même simple comme nano, peut apporter quelques facilités dans l'écriture de programmes.

2.1 Coloration syntaxique

La coloration syntaxique signifie utiliser des couleurs pour mettre en évidence certaines parties d'un code : mots clés, constantes...

Pour utiliser cette facilité, il faut configurer NANO. Cette configuration se fait dans le fichier `~/.nanorc`.

Tutoriel 2

Introduire la coloration sytaxique

- Tapez `nano ~/.nanorc` pour éditer le fichier de configuration de *nano*.
- Ajoutez-y la ligne : `include "/usr/share/nano/java.nanorc"`
- Quittez l'éditeur.
- Ouvrez le fichier `Ex.java` ; il devrait être coloré.

2.2 Numérotation des lignes

Nano peut également indiquer le numéro de la ligne sur laquelle se trouve le curseur, ce qui sera pratique pour corriger vos erreurs. Pour cela, il existe deux méthodes.

- ▷ **Méthode 1** : Lancer nano avec l'option `-c` : `nano -c monFichier`.
- ▷ **Méthode 2** : Dans l'éditeur, appuyez sur `CTRL-c`.

2.3 Indentation

Pour qu'un programme soit lisible, il doit être *indenté*. Ce qui serait pratique lorsqu'on code ce serait qu'un retour à la ligne positionne automatiquement le curseur de façon à être aligné avec la ligne précédente. Pour que nano fasse ça pour nous, il suffit d'ajouter ceci à son fichier de configuration : `set autoindent`.

2.4 Autres configurations

Si vous désirez connaitre d'autres possibilités de configuration de nano, vous pouvez lire le manuel : `man nanorc`.

Pour une *quick ref* en ligne, consultez (par exemple) :
www.codexpedia.com/text-editor/nano-text-editor-command-cheatsheet/ .

3 Compiler et exécuter du Java

Java sans package

Si le programme Java n'utilise pas de package :

- ▷ `javac MaClasse.java` compile le programme Java du fichier donné
- ▷ `java MaClasse` : exécute le programme Java se trouvant dans la classe donnée.

Tutoriel 3

Compiler / exécuter un programme

Commençons par un programme correct et tentons de l'exécuter.

Le fichier `/eCours/java/td/td3/Ex.java` contient un petit programme Java tout simple qui affiche un message de bienvenue.

- Si ce n'est pas le cas, placez-vous dans le dossier `td3`.
- Copiez le fichier indiqué dans cd dossier : `cp /eCours/java/td/td3/Ex.java .`
- Lisez-le et voyez si vous devinez ce qu'il fait : `cat Ex.java`
- Compilez-le : `java Ex.java`
- Que fait cette phase, à quoi sert-elle ? Affichez le contenu du dossier pour le vérifier.
- Exécutez le programme : `java Ex`

Lorsqu'on compile, il faut mettre le `.java` mais lorsqu'on exécute, il ne faut pas mettre le `.class`.

Retenez !
On **compile un fichier** mais on **exécute une classe**.

Vous allez à présent écrire votre premier programme de bout en bout sur linux1.

Exercice 3

Un programme Java

Écrivez un programme qui :

- ▷ Affiche votre nom ;
- ▷ Affiche la valeur de 7^2 ;
- ▷ Affiche la valeur de 2^7 ;
- ▷ Demande une entier à l'utilisateur et affiche s'il est pair.

Conseils
Procédez par étape et vérifiez correctement votre programme. Ce n'est pas parce qu'il compile qu'il est correct. Et ce n'est pas parce qu'il affiche quelque chose que c'est la bonne réponse.

Exercice 4 **Comprendre les erreurs (I)**

Supposons que vous faites une erreur dans votre programme. Par exemple en écrivant `Public` au lieu de `public`.

À quelle étape le problème va-t'il apparaitre ? Sous quelle forme ? Testez !

Exercice 5 **Comprendre les erreurs (II)**

Supposons que vous demandez un calcul impossible dans votre programme. Par exemple `1/0`.

À quelle étape le problème va-t'il apparaitre ? Sous quelle forme ? Testez !

4 Transfert de fichiers

Vous n'avez peut-être pas fini. Pour pouvoir continuer à la maison sans tout recommencer, il serait bon de pouvoir récupérer ce que vous avez déjà fait sur linux1. Voici une façon de le faire.

Tutoriel 4 **Transférer des fichiers**

- Ouvrez l'explorateur de fichier Windows (par exemple en cliquant sur l'icône "My Computer").
- Dans le champ d'adresse, tapez l'adresse `ftp://linux1`.
- Une boite de dialogue vous demande votre login et mot de passe (sur *linux1*).
- Vous voyez apparaitre votre dossier personnel sur *linux1*.
- Vous pouvez y prendre/déposer des fichiers comme vous le feriez pour un dossier Windows. Vous pouvez par exemple les mettre sur une **clé USB**, sur le cloud ou vous les envoyer par mail.

5 Conclusion

5.1 Ce qu'il faut connaitre

Notions importantes de ce TD

Voici les notions importantes que vous devez avoir assimilées à la fin de ce TD.

- ▷ La notion de fichier caché et comment les voir.
- ▷ Savoir utiliser `nano` pour éditer un petit programme Java.
- ▷ Savoir compiler et exécuter un programme Java qui n'utilise pas de package.

5.2 Pour aller plus loin : l'éditeur `vi`

Si vous avez fini, vous serez peut-être curieux d'apprendre à utiliser `vi` plutôt que `nano`.

Quels sont les avantages de `vi` ?

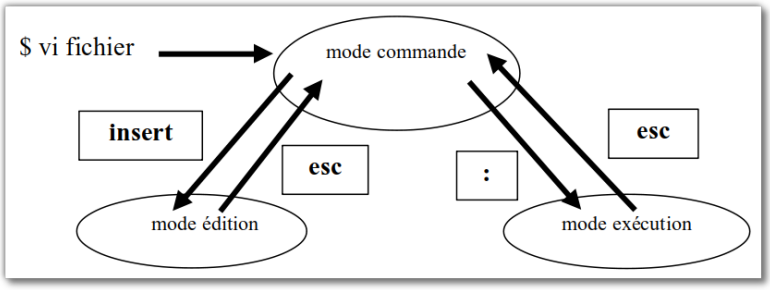
- ▷ Certaines manipulations du fichiers sont plus simples : copier, supprimer, déplacer des lignes par exemple.
- ▷ Il est facile d'indenter proprement un programme Java qui ne l'est pas.

`vi` est plus puissant que `nano` mais il est moins intuitif quand on débute. Voici les bases à comprendre.

Démarrer

Pour éditer un fichier texte existant ou créer un nouveau fichier, il suffit de taper : `vi monFichier`.

Les 3 modes de fonctionnement sous vi(m) :



Le mode commande

C'est le mode dans lequel vous vous trouvez quand vous ouvrez `vi`. Dans ce mode, les touches sur lesquelles vous appuyez ne sont pas insérées dans le texte (comme dans nano) mais sont considérées comme des commandes. C'est le mode qui permet de **manipuler** le texte.

Exemples de commandes :

- ▷ `i` : passer en mode édition (cfr. infra) ;
- ▷ `yy` : copier la ligne sous le curseur (comme un `CTRL-C` sous Windows) ;
- ▷ `3yy` : copier 3 lignes ;
- ▷ `dd` : couper la ligne sous le curseur (`5yy` pour en couper 5) ;
- ▷ `dw` : couper le mot sur lequel se trouve le curseur ;
- ▷ `p` : coller (ce qui a été précédemment copié ou coupé) sous la ligne qui suit le curseur ;
- ▷ `u` : annuler la dernière modification. Vous pouvez appuyer plusieurs fois pour annuler les dernières modifications.

Le mode édition

C'est le mode dans lequel ce qu'on tape est ajouté au texte, comme dans nano. On y accède par la touche :

- ▷ `i` (insert) : on insère à l'endroit du curseur ;
- ▷ `a` (append) : on insère après le curseur ;
- ▷ `o` : on insère dans une nouvelle ligne créée sous le curseur ;

Dans tous les cas, l'indicateur `INSERT` apparait alors en bas de l'écran.

Le mode exécution

On y accède à partir du mode commande en tapant `:`. Ce mode permet d'entrer d'autres types de commandes, plus riches que celles du mode commande. Voici les plus utilisées :

- ▷ `:h` pour accéder à l'aide (`q` pour quitter l'aide) ;
- ▷ `:w` pour sauver le fichier ;
- ▷ `:w monFichier` pour enregistrer sous `monFichier`,
- ▷ `:q` pour quitter l'éditeur (sans sauver) ;
- ▷ `:x` pour enregistrer et quitter ;
- ▷ `:q!` pour quitter sans enregistrer les modifications ;
- ▷ `:set nu` pour afficher les numéro de ligne (`:set nonu` pour les retirer) ;
- ▷ `:numéroDeLigne` pour aller directement à cette ligne ;
- ▷ `:%s/old/new/g` pour remplacer toutes les occurrences de la chaine de caractères `old` par la chaine de caractère `new`.