

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

1	Quelques notions supplémentaires de Linux	2
1.1	La ligne de commande	2
1.2	Les fichiers cachés	3
2	L'éditeur nano	3
2.1	Coloration syntaxique	3
2.2	Numérotation des lignes	3
2.3	Indentation	3
2.4	Autres configurations	4
3	Java en mode console	4
3.1	Compiler et exécuter	4
3.2	Produire la Javadoc	5
4	Les variables d'environnement	6
4.1	Introduction	6
4.2	Le prompt	6
5	Transfert de fichiers	7
6	Conclusion	7
6.1	Ce qu'il faut connaître	7
6.2	Pour aller plus loin : l'éditeur vi	7

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

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 de 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 `monfichier` au nom tellement long qu'il me paraît peu probable de le taper 2x sans erreur 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-exécute la commande de numéro donné.
- ▷ `!cmd` : ré-exécute 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 syntaxique

- ✍ 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 connaître 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 Java en mode console

Avec Netbeans, lorsque vous appuyez sur des boutons (comme celui demandant d'exécuter le projet), cela lancent des commandes qui effectuent la tâche. Il est possible d'exécuter directement ces commandes. C'est ce que nous allons faire ici.

3.1 Compiler et exécuter

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/dev1/env1/Hello.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/dev1/env1/Hello.java .`
- ✍ Lisez-le et voyez si vous devinez ce qu'il fait : `cat Hello.java`
- ✍ Compilez-le : `java Hello.java`
- ✍ Que fait cette phase, à quoi sert-elle ? Affichez le contenu du dossier pour le vérifier.
- ✍ Exécutez le programme : `java Hello`

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 ;
- ▷ Demande un nombre à l'utilisateur ;
- ▷ Affiche l'inverse du nombre (par ex : 0.25 si le nombre entré était 4).

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 apparaître ? 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 apparaître ? Sous quelle forme ? Testez !

3.2 Produire la Javadoc

Javadoc

`javadoc -doc dossierDoc MaClasse.java` génère la javadoc présente dans le fichier `MaClasse.java` dans le dossier `dossierDoc` (qui doit exister).

Exercice 6

Javadoc de Hello

Produisez la javadoc du programme `Hello` précédent.

Visualiser le résultat

La javadoc est au format HTML et peut être consulté avec un simple navigateur web mais il n'y en a pas sur `linux1`¹.

TODO

À voir s'il y aura un serveur apache sur le nouveau `linux1`. Si oui, on parlera du `public_html`, sinon parlera de FTP.

1. Un Linux avec environnement graphique dispose bien sûr d'un navigateur web.

4 Les variables d'environnement

4.1 Introduction

Variable d'environnement

Une **variable d'environnement** est une variable associée à votre shell contenant un texte qui est accessible par toutes les applications que vous lancez. Généralement, elle permet de configurer certaines applications, d'en modifier le comportement.

Manipuler une variable d'environnement

Si VARENV est une variable d'environnement :

- ▷ `export VARENV=valeur` crée une nouvelle variable à la valeur donnée.
- ▷ `VARENV=valeur` modifie la valeur d'une variable existante.
- ▷ `$VARENV` est remplacé par la valeur de la variable.

Experience 1

Manipuler une variable

- ✍ Entrez `VAR=12` pour créer la variable.
- ✍ Entrez `echo "Bonjour !"`.
Cette commande affiche ce qu'on lui donne en paramètre.
- ✍ Entrez `echo $VAR` pour afficher le contenu de la variable.

4.2 Le prompt

Prompt

Le **prompt** (ou *invite* en français) est le texte qui apparaît à gauche de ce que vous tapez dans votre shell. Il est déterminé par la variable d'environnement `PS1`.

Experience 2

Le prompt

Affichez la valeur de votre prompt. Vous remarquerez qu'il contient des codes qui seront remplacés par certaines valeurs. Par exemple, `\w` indique le dossier courant.

Exercice 7

Modifier le prompt

Modifiez la valeur de votre prompt. Par exemple, modifiez l'invite en "Bonjour! ".

Tutoriel 4

Durée de vie de la variable

- ✍ Déconnectez-vous de `linux1`, puis reconnectez-vous.

Surprise, votre prompt a repris sa valeur d'origine ! Pour rendre une modification permanente, il faut ajouter la commande à votre fichier `.bashrc`. C'est un fichier caché qui est exécuté par votre shell lors de votre connexion.

Exercice 8

Explorer les codes du prompt

Les possibilités de configuration du prompt sont nombreuses : afficher l'heure, le nom de la machine, votre login, utiliser des couleurs... Examinez la documentation pour configurer le prompt comme il vous plait (`man bash`, section `PROMPTING`). Rendez la modification permanente.

5 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 5

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 boîte de dialogue vous demande votre login et mot de passe (sur `linux1`).
- ✍ Vous voyez apparaître 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.

6 Conclusion

6.1 Ce qu'il faut connaître

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.
- ▷ Savoir créer/consulter/modifier une variable d'environnement.

6.2 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` ?

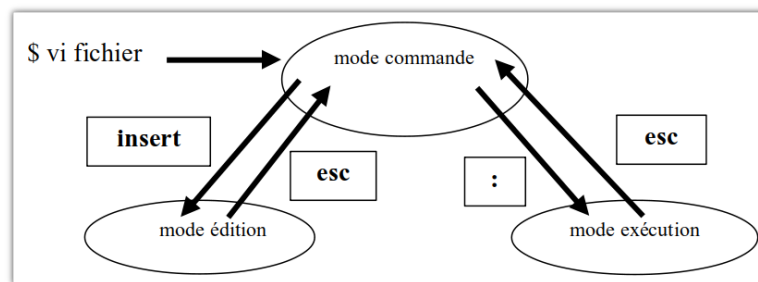
- ▷ Certaines manipulations du fichiers sont plus simples : copier, supprimer, déplacer des lignes par exemple.
- ▷ Il est facile d'indenter proprement un programme Java qui ne l'est pas.

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** apparaît 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 chaîne de caractères `old` par la chaîne de caractère `new`.