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

2019 – 2020
DEV1 – ENVL

DEV1 – ENVL – Laboratoires d'environnement

TD 0 – Lost material

Table des matières

1 Rechercher (dans) des fichiers

Lorsqu'on commence à avoir beaucoup de fichiers, il arrive qu'on ne sache plus trop bien où se trouve un fichier ou quel fichier contient un texte particulier. Les commandes `find` et `grep` sont là pour nous aider.

1.1 Rechercher un fichier

La commande `find` permet de chercher des fichiers dans le système de fichiers en fonction de critères : son nom, sa taille, sa date de création...

find

`find dossierOùChercher critèreDeRecherche...` cherche dans le dossier indiqué (et tous les sous dossiers) les fichiers qui respectent tous les critères indiqués. Par défaut, la commande affiche le nom des fichiers trouvés.
Parmi les critères citons :

- `-name nomfichier`

2 Les variables d'environnement

Version du document
20 juin 2019

2.1 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.

Exercice 1 Comprendre la signification du $

Supposons que la variable d'environnement `VAR` vaut 12. Que va faire chacune des commandes suivantes ? Une fois que vous pensez le savoir, vérifiez-le le tapant sur la machine.

- ▷ `echo VAR`
- ▷ `$VAR=12`
- ▷ `VAR=$VAR`
- ▷ `echo $VAR + 30`
- ▷ `VAR=$VAR+30`
- ▷ `VAR=$VAR + 30`
- ▷ `VAR="$VAR + 30"`
- ▷ `VAR=VAR`

2.2 Le prompt

Prompt

Le **prompt** (ou *invite* en français) est le texte qui apparait 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 2 **Modifier le prompt**

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

Tutoriel 1 **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.

2.3 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` ?

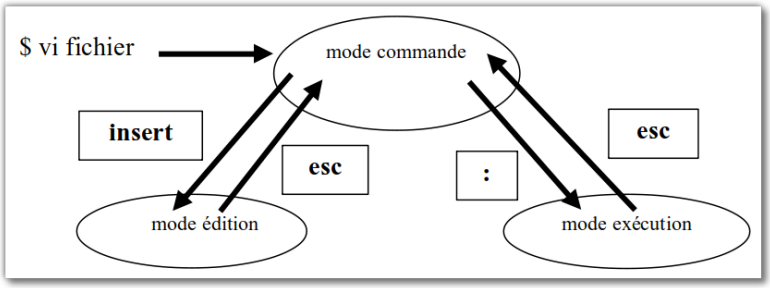
- ▷ Certaines manipulations du fichiers sont plus simples : copier, supprimer, déplacer des lignes par exemple.
- ▷ Il est facile d'indenter proprement un programme Java qui ne l'est pas.

`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`.

2.4 Pour aller plus loin : le prompt

Exercice 3 **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.