TP n°0 de Programmation 1 : L'environnement de travail

François Thiré

September 14, 2016

L'objectif de cette session est de mettre en place un environnement de travail afin de programmer dans de *bonnes* conditions. En effet, les programmes que vous serez amenés à développer par la suite seront beaucoup plus complexes que ce que vous avez pu faire jusqu'à maintenant (enfin je suppose...). Il faut donc prendre les bonnes habitudes le plus tôt possible.

L'environnement de travail se décompose en plusieurs outils que vous devrez utiliser pour éxécuter vos propres programmes. Cependant il n'est pas possible (ni utile) de maîtriser complètement ces outils. Votre apprentissage se fera progressivement au cours de vos études.

Pour ce cours, l'objectif va être de vous faire utiliser d'abord un peu la ligne de commande avec BASH. Puis d'utiliser un *vrai* éditeur de texte comme EMACS. Enfin, on verra comment compiler un programme écrit dans le langage OCAML[1].

Si vous avez la moindre question en dehors des TP, vous pouvez me contacter à l'adresse `francois.thire@lsv.ens-cachan.fr`.

1 La ligne de commande avec BASH

BASH est l'outil qui va nous permettre de communiquer avec le *système d'exploitation*. En 411, les ordinateurs utilisent un système d'exploitation nommé UBUNTU qui se base sur un *noyau* qui est appelé GNU/LINUX (ou ou tout simplement LINUX). Vous verrez dans le cours *Architectures & Systèmes* ce que signifie plus en détail les termes *système d'exploitation* et *noyau*.

La ligne de commande nous permet de donner des ordres au système d'exploitation. C'est une alternative à l'interface graphique que vous utilisez avec la souris (même si historiquement l'interface graphique est venue après la ligne de commande). Dans notre cas, la ligne de commande s'avèrera beaucoup plus pratique[2].

Ouvrir la ligne de commande : Ouvrez le menu *Applications* puis aller dans *Accessoires* et enfin cliquer sur *Terminal*.

Un écran noir s'affiche et normalement vous devriez voir quelque chose de la forme

```
fthire@zip:~$
```

le symbole @ sépare votre nom d'utilisateur du nom de la machine. Dans mon cas `fthire` est mon nom d'utilisateur et `zip` est le nom de la machine. Les : séparent le nom de la machine du répertoire courant, ici `~`. Le tilde est un raccourci pour désigner le *home*. C'est dans ce répertoire que vous enregistrerez tous vos fichiers.

Vos premières commandes : On va tester une première commande : `ls` (*list*). Cette commande a pour effet de lister les fichiers qu'il y a dans le répertoire courant. À partir de maintenant, une commande sera typographiée de la façon suivante

```
$ ls
```

Vous devriez voir alors s'afficher quelque chose de la forme

```
Desktop Documents Downloads Music Pictures Public Videos
```

La commande *ls* affiche en bleu les *dossiers*. Si vous voulez changer de dossier, vous pouvez utiliser la commande `cd` (*change directory*). La commande `cd` prend un paramètre qui est le nom du dossier comme ceci :

```
$ cd Documents
```

[1] Pour ceux qui ont fait du CAML-LIGHT vous ne serez pas trop dépaysés

[2] Faites-moi confiance sur ce coup!

Le BASH fournit une fonctionnalité qui s'appelle l'autocomplétion. Cette dernière vous permet de ne pas avoir à entrer entier le nom du dossier complet. Essayez la commande suivante par exemple

```
$ cd Doc<TAB>
```

Vous pouvez aussi observer que le ~ s'est changé en `~/Documents`. Le BASH vient par défaut avec quelques centaines de commandes. Èvidemment, il n'est pas possible de les retenir toutes! Pour cela, il existe la commande `man`. Cette commande prend en paramètre une autre commande. En lançant une commande *man*, vous changez de mode. Pour quitter le manuel, il vous faudra appuyer sur la touche `q`. Essayez par exemple la commande suivante :

```
$ man ls
```

Enfin, il me reste à vous parler des *options*. Les options permettent de modifier le comportement d'une commande. Par exemple, essayez la commande

```
$ ls -l
```

Pour savoir ce que fait cette option précisément, allez regarder le manuel (dans le jargon, on emploie généralement l'acronyme RTFM pour *inviter* une personne à lire le manuel, je vous laisse deviner ce que ça veut dire...).

On distingue généralement deux types d'options. Les options *courtes* et les options *longues*. Les options courtes font un seul caractère et sont introduites par le caractère `-`. C'est le cas par exemple de l'option `l` que l'on vient de voir. Les options longues font plusieurs caractères et sont introduites par `-`. Par exemple

```
$ ls -l -human-readable
```

Question 0 : Entrez la commande `$ ls -a` . Parmi les nouvelles entrées qui apparaissent, il y en a deux qui sont étranges : `.` et `..` . Essayez de comprendre leur particularité (en utilisant la commande `cd` par exemple).

1.1 Votre espace de travail pour ce cours

Pendant les TPs, vous allez créer de nombreux fichiers. Afin d'éviter qu'ils soient rangés n'importe comment, on va créer des dossiers pour ranger vos fichiers. Je vous invite donc à adopter la structure suivante pour les fichiers que vous créerez pour ce TP : `Cours/L3/Programmation-1/TP0`.

On n'a pas encore vu comment créer un dossier dans le terminal. Vous aurez certainement besoin des commandes `mkdir` et `rmdir`.

Question 1 : En utilisant le manuel, créez l'arborescence de dossiers suggérée ci-dessus. N'hésitez-pas à vour servir du manuel. En bonus, essayez de créer ces dossiers en une seule commande.

A l'issue de cette section, vous êtes censés capable de naviguer à travers le système de fichiers de Linux en utilisant les commandes du terminal. On va voir maintenant, comment créer et éditer des fichiers en utilisant un éditeur de texte.

2 L'éditeur de texte

L'éditeur de texte va être l'outil avec lequel vous allez éditer tous vos fichiers. Que ce soit pour programmer, rédiger un rapport ou bien même éditer la configuration d'un logiciel. Il faut bien le choisir car vous allez probablement vous en servir pendant des dizaines d'années!

Cependant, il n'éxiste pas un éditeur de texte universel, et choisir son éditeur de texte favori est souvent une question de goût. Parmi les éditeurs de texte les plus connus il y a

- Emacs
- Vi
- Gedit
- Sublime Text
- Atom

En général, le choix se fait surtout entre Emacs et Vi. Ces deux éditeurs ont les avantages d'être très répandus et extrêmement personnalisables. De mon côté, j'utilise Emacs, et je pourrai vous aidez seulement avec cet éditeur. C'est d'ailleurs ma recommandation personnelle.

Cependant, il peut-être intéressant pour vous de tester les deux éditeurs afin de faire votre choix. D'autant plus qu'il est intéressant de connaître le fonctionnement basique de ces deux éditeurs puisque certains raccourcis de ces éditeurs sont utilisés par le BASH et certaines de ses commandes.

Emacs : Pour apprendre à utiliser Emacs, vous pouvez lire le tutoriel (ce qui prend environ 30mn). Pour cela entrez la commande

```
$ emacs
```

puis appuyer sur la touche `F1` et ensuite la touche `t`. Cela lancera le tutoriel pour vous, et vous n'aurez plus qu'à le lire. Vous pouvez aussi jeter un coup d'oeil au document suivant http://www.jesshamrick.com/2012/09/10/absolute-beginners-guide-to-emacs/.

Vi : Pour Vi, vous pouvez aller voir sur ce site http://www.openvim.com/ qui propose un tutoriel interactif.

A partir de maintenant, je suppose que vous avez choisi votre éditeur de texte. On va voir comment *compiler* notre premier fichier Ocaml.

3 La compilation avec Ocaml

La compilation est un processus qui transforme un fichier écrit dans un langage A en un autre fichier écrit dans un langage B. La machine que vous utilisez ne reconnaît qu'un seul langage qui est un langage binaire. Lorsque vous voulez que votre machine éxécute du code Ocaml, vous allez donc devoir compiler votre fichier écrit en Ocaml vers un fichier qui contient du code binaire. Heureusement, le programme qui fait ce travail a déjà été écrit pour vous. Ce programme est une commande comme *ls* qui s'appelle *ocamlc*. Elle s'utilise de la façon suivante:

```
$ ocamlc <FILE>
```

Question 2 : Une fois dans le dossier `TPO`, utiliser la commande touch pour créer un fichier vide. Comme ce fichier est un fichier Ocaml, on va lui donner une extension pour que ce soit plus facile à repérer. Les fichiers Ocaml ont pour extension *.ml* (on vera plus tard qu'il existe aussi des fichier *.mli*. Puis avec la commande *ocamlc* compiler un fichier vide. Quel est le résultat de cette commande ? Lisez le manuel de la commande *ocamlc* pour trouver comment donner un nom au fichier produit.

Pour exécuter le fichier produit par la commande *ocamlc* vous pouvez utiliser

```
$ ./a.out
```

par exemple. En réalité, vous invoquez `a.out` comme si c'était une commande normale en lui donnant sont chemin relatif. Le fichier étant vide, évidemment ce programme ne fait rien. Cependant, si vous faites un `$ ls -l` , le fichier obtenu n'est pas vide. Il contient en fait un certains nombre d'informations qui permettent au système d'exploitation d'*éxécuter* le fichier.

Et maintenant si on écrivait un vrai programme Ocaml? Je vous invite donc à ouvrir un fichier *hello.ml* dans lequel vous allez écrire

```
let _ = Printf.printf "Hello World!\n";
```

et si vous le compilez, vous devriez voir dans la console un magnifique *Hello World!* s'afficher. Vous venez de créer votre premier programme Ocaml!

4 Ressources diverses

Cette section est à regarder après avoir fini ce TP.

Dans le prochain TP, on va attaquer la programmation en Ocaml. Il se peut qu'une minorité d'entre-vous n'ayant pas fait de CAML-LIGHT soit désemparée face à ce nouveau langage. La toile regorge de ressources diverses et variées pour apprendre Ocaml. Je vous invite donc à lire les différentes ressources sur Ocaml qui sont citées ci-dessous (ou d'autres selon vos goûts).

4.1 Premiers pas en Ocaml

Pour ceux qui découvrent le langage Ocaml, je vous invite à commencer par lire les 8 premiers chapitres de ce tutoriel http://mirror.ocamlcore.org/ocaml-tutorial.org/. Sinon, un rapide résumé a été fait par Sylvain Schmitz et est accessible à l'adresse suivante : http://www.lsv.ens-cachan.fr/~schmitz/teach/2012_prog/part1/td1_ocaml/caml.pdf. Une partie des exercices de cette feuille seront repris dans le prochaine TP.

4.2 Aller plus loin avec la ligne de commande

A l'adresse suivante : http://overthewire.org/wargames/bandit/, vous trouverez un petit jeu découpé en niveau qui vous demande de trouver la bonne commande pour passer au niveau suivant. Cela vous fera un bon entraînement pour apprendre à lire le manuel des commandes BASH. Attention, certains niveaux peuvent être corsés.

BASH est en réalité un langage de programmation. Il est possible d'écrire des scripts en BASH. Pour apprendre à programmer en BASH, vous pouvez par exemple lire les premières pages du manuel officiel : https://www.gnu.org/software/bash/manual/bash.pdf.

4.3 Aller plus loin avec Emacs

- Le manuel d'emacs : https://www.gnu.org/software/emacs/manual/pdf/emacs.pdf
- Une introduction à emacs-lisp : https://www.gnu.org/software/emacs/manual/pdf/eintr.pdf
- Une *cheatsheet* des raccourcis claviers : https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

4.3.1 Personnaliser Emacs

La personnalisation d'Emacs se fait en ajoutant des modes. En voici quelqu'uns qui pourront vous être utiles.

- Merlin (pour Ocaml) : https://github.com/the-lambda-church/merlin/wiki/emacs-from-scratch.
- Rainbow-Mode