Programmation éditoriale
I. Introduction à Unix / Linux

Éric Guichard*

1er juin 2009

Ce document est avant tout un aide-mémoire pour les personnes qui s'initient à Linux et à la programmation : il accompagne un cours de programmation éditoriale (à l'ENSSIB), devant des machines, et n'a pas vocation à s'y substituer.

Il propose aussi une mise en perspective de l'informatique et de la programmation, au regard de la culture écrite : la plupart de mes étudiants sont des «littéraires», et je crois que les passerelles entre le monde des programmeurs et les mondes lettrés de la période de l'imprimé sont plus nombreuses et variées qu'on ne l'imagine.

De toute façon, même en admettant qu'ils soient distincts, ces deux mondes ont un point commun : ce n'est pas avec un seul mode d'emploi, une seule carte qu'ils se laissent explorer. Et si ces quelques pages ont une quelconque utilité, ce ne pourra être qu'en complément aux milliers ou millions d'autres pages, livres, articles, essais... qui proposent des cours, des conseils, des critiques sur des thèmes analogues à ceux abordés dans ce document.

Note : la table des matières est à la fin du document.

1 Le projet initial

Cette introduction replace le cours dans son contexte géographique, pédagogique, et temporel (Lyon, 2009, pour des volontaires de culture littéraire) .

1.1 Rappel du descriptif

Le propos de cet enseignement est d'apprendre à réaliser un réel *logiciel en ligne*, ce qui me semble le mode ultime de l'édition de pages dynamiques et un passage obligé pour toute forme de publication numérique un peu complexe.

*Maître de conférences à l'ENSSIB, responsable de l'équipe *Réseaux, Savoirs & Territoires* de l'École normale supérieure.

Une telle production alimente la réflexivité : comment la réalisation technique d'un site web et une problématique éditoriale s'infléchissent-elles l'une l'autre ? comment l'une et l'autre facilitent-elles la compréhension de ce qu'est une technologie de l'intellect, Comment la pratique (mettre les mains dans le «cambouis des octets») permet-elle de s'affranchir des fausses questions quant à l'internet et de construire à son sujet des problématique pertinentes ?

1.2 Fil conducteur du cours

Je rappelle ici quelques points développés oralement.
- Enseignement ouvert à tous (ENSSIB ou non, étudiants ou non).
- Accompagnement bienveillant de tous. Donc accepter vitesses variables.
- Culture de l'écrit contemporain. Cf. érudition XVI^{e} siècle, XIX^{e} siècle, XXI^{e} siècle. Et littératie, toujours contextualisée socialement, culturellement, *ad hoc*.
- Non pas des langages, mais des lexiques et des grammaires sommaires. Savoirs diffusés de façon traditionnelle, depuis l'invention de l'écriture : grâce à autrui.
- Évolution : cgi-bin (*Common Interface Gateway*), SVG (*Scalable Vector Graphics*), LaTeX vers html, BibTeX vers XML (*eXtensible Markup Language*).

1.3 Exemples de réalisations

- http://barthes.enssib.fr/atlasclio/
- http://barthes.enssib.fr/reptraqueur

2 Linux / Unix

2.1 Apports

Une approche presque épistémologique.
- Pemier intérêt : se familiariser avec un autre OS (*Operating System*).
- Mais aussi avec les ressources de Windows et du Mac (ce dernier OS est construit à partir d'un noyau Linux) : les ordinateurs fonctionnant de manière analogue, comprendre les arcanes de Linux aide à comprendre les autres OS.
- L'informatique perd de sa magie pour redevenir une technique efficace fabriquée par les humains pour les humains.
- Notion de brique logicielle : Cf. 3.4. Abandonner les outils imposés pour retrouver la liberté de créer.
- Et surtout : entrer de plein pied dans le monde des robots de l'internet.

2.2 Une présentation iconique

Pour le profane, l'ergonomie Linux ne se distingue plus de celle des autres OS :

- Menus de l'OS : pomme, chapeau, Windows...
- Un *bureau* toujours visualisé d'emblée.
- Lancement automatique des logiciels *ad hoc* : double-clic sur une icône.
- Etc. Il n'y a donc plus aucune raison d'avoir peur de Linux.

2.3 Le terminal

Outil essentiel. Sobre, voire laid, parfois impoli (*command not found*), parfois bavard, parfois (trop ?) silencieux.

— *Définition* : Un terminal (ou émulateur de terminal) est une interface de communication en l'humain et le système d'exploitation. Il permet de saisir et d'afficher du texte, souvent adapté à un interpréteur de commandes (qu'on appelle un *shell*), sinon à des programmes élaborés : un éditeur de texte comme `joe`, un logiciel pour gérer ses mails comme `mutt`, un processeur de texte comme LaTeX, un navigateur comme `lynx`, un programme comme `perl`, etc.

On peut tout faire (et très vite) avec un terminal. C'est aussi le principal moyen pour communiquer avec un ordinateur distant.

Principaux équivalents : `putty`, sinon `cmd` sur Windows. `Terminal` (Applications -> Utilitaires -> Terminal.app), sinon `X11` sur un Mac.

2.4 Architecture et chemins d'accès

Ce point informatique est désormais bien connu de tous. Pour Linux / Unix, il suffit de se rappeler que l'articulation d'un chemin d'accès est le `/`. Le chemin d'accès absolu à un fichier, à un dossier, ou à une commande est donné par la succession des dossiers qui y conduisent, en partant de la racine (*root*) notée `/`.

Exemples : `/usr/bin/perl`, `/home/dupont/pancarte.jpg`.

On peut aussi parcourir l'arborescence Unix de façon relative : `..` pour remonter d'un cran, ∼ pour passer «chez» un utilisateur (`cd ∼dupont`), etc.

3 Quelques commandes Unix

Je conseille une (re)lecture non linéaire de ce paragraphe : certains points relatifs à la culture seront mieux compris après l'expérimentation de commandes, les uns et les autres seront mieux assimilés après une vraie pratique et la consultation de plusieurs autres documentations, etc.

3.1 Culture : astuces et raccourcis

Les savoirs qui suivent peuvent apparaître parfois élémentaires, mais ils relèvent de la littératie de base de l'informatique, sinon de la bouée de sauvetage ; et nous pourrions montrer que la culture classique et imprimée sollicite elle aussi de telles connaissances, euphémisées voire oubliées tellement elles sont présentes et indispensables.

3.1.1 Dans une fenêtre de terminal

- Complétion d'un nom de fichier, de dossier ou d'une commande (touche tabulation).
- Rappel des commandes antérieures (avec les flèches montantes et descendantes du clavier). Cf. aussi la commande `history`.
- `Control-C` (maintenez appuyée la touche de fonction Control, et appuyez une fois sur la touche D, puis relâchez les deux) ou `Ctrl-D` : pour arrêter un processus en cours, ou pour **se sortir d'un mauvais pas**.
- `q` (quitter) pour clore le `less` (et bien d'autres logiciels).
- Manuel : la plupart des commandes et programmes que vous utilisez avec le terminal disposent d'une aide, (cf. point 3.3).
- Audace : Osez ouvrir tout type de fichier avec un éditeur (`nedit`, `kwrite`, `joe`, `emacs`, etc.) ou avec l'afficheur `less`.

3.1.2 Général

Ce point vaut pour la majorité des logiciels et oscille entre entre bon sens et conseils.

- *Raccourcis clavier* (avec la touche Ctrl) : copier, coller, imprimer, rechercher, etc.
- *Copier-coller rapide* : tout texte sélectionné (en inversion vidéo) est glissé dans le presse papier. Pour le coller, il suffit de cliquer avec les deux boutons de la souris (à la fois) à l'endroit voulu.
- Évitez de saisir les chiffres de votre *login* ou de votre mot de passe avec le pavé numérique : celui-ci n'envoie pas toujours des chiffres tant que votre session n'est pas effectivement ouverte (vos paramètres ne sont pas encore «lus» par le système).
- *Organisation des fenêtres* : vous allez vite travailler avec une dizaine de fenêtres à la fois ; évitez que certaines prennent tout l'écran, et ajustez-les pour que leurs bords figurent un escalier. Ainsi, vous pourrez aisément sélectionner celle dont vous aurez besoin. En bas de l'écran, une représentation miniaturisée de ces fenêtres est souvent affichée.
- *Encodages* : ne vous inquiétez pas si, au début, les accents ne correspondent pas du tout à ce que vous attendez. Ce point déroute ou énerve les néophytes,

il se règle en fait assez aisément. Consultez http://www.tuteurs.ens.fr/faq/utf8.html si vous êtes pressé/e d'en savoir plus.
– *Messages d'erreur* : n'hésitez pas à insérer toute réponse impolie de l'ordinateur dans une requête à un moteur de recherche. Souvent, la réponse vous satisfera.

3.2 Documentations

Dans le monde de l'imprimé, *évitez* les traductions, souvent truffées d'erreurs (cas le moins tragique : un $ devient un S, etc.).

3.2.1 Sur l'internet

Voici une petite sélection :
– http://www.linux-france.org/article/debutant/debutant-linux.html
– http://www-inf.enst.fr/~danzart/fiches/unix_abrege.html
– http://www.tuteurs.ens.fr/unix/

3.2.2 Sur la machine

On découvre plusieurs documentations en passant des commandes dans le terminal. Elles ne sont pas toujours très conviviales ; parfois elles sont traduites en français. Pour quitter ces *pages* d'explications, on tape `q` (on en sort comme on sort du `less`).

La plus importante est la commande `man` (*manual*). Par exemple, `man man` explique comment fonctionne ce manuel. `man lacommande` détaille l'usage de la commande *lacommande*. Exemples : `man less`, `man mv`, `man chown`.

On peut aussi rechercher dans quel contexte, est utilisé un mot ou une commande :
`man -k lacommande`. Cela peut être utile lorsque le nom de la commande n'est pas intuitif. Ex. : `man password` ne donne pas de réponse, car la commande permettant de changer de mot de passe ne s'intitule pas *password*. Mais un `man -k password` aide à comprendre que la commande recherchée est `passwd`.

D'autres questions ? Voir http://www.tuteurs.ens.fr/unix/aide.html

3.3 Commandes les plus utiles

Les commandes doivent être saisies dans un terminal et conclues par un passage à la ligne pour devenir effectives (le retour-chariot signale au système la fin

de la commande).

Une commande est souvent suivie d'options (précédées d'un tiret) et de paramètres. Exemple : `ls -l *jpg` : affiche de façon détaillée (option `-l`) les descriptifs de tous les fichiers se terminant par `jpg` (paramètre `*jpg`).

- `ls` (list) et ses variantes (options) :
  - `ls -l` (utile pour connaître les propriétaires des fichiers, les droits d'exécution...).
  - `ls -la` (liste tous les fichiers, y compris les cachés [ceux dont le nom commence par un point]).
  - `ls -lRt` (récursion + ordre temporel inverse)
- `pwd` (où suis-je ?). Utile pour distinguer les zones Unix où l'on risque fort d'être (chez soi, zone éditoriale, cgi-bin...)
- `mkdir` (make directory). `mkdir folder` crée un dossier du nom de *folder* en dessous de l'endroit où on est. Voir aussi : `rmdir, rm -r, rm -rf, mv, cp`
- `less` (affiche page d'écran par page d'écran le contenu d'un fichier). Exemple : `less file` affiche le contenu du fichier dont le nom est *file*.
  Quand on est dans le *less*, les touches suivantes ont les effets suivants :
  - `q` : quitter l'affichage (quitte l'application *less*) .
  - *barre d'espacement* : descendre d'une page.
  - *retour-chariot* : descendre d'une ligne.
  - `k` : remonter d'une ligne.
  - `g` : remonter en première page.
  - `G` : descendre à la dernière page.
  - `/expr` : recherche l'expression *expr* dans le document.
- `chmod` (change mode). Très utile pour que le programme CGI soit rendu exécutable (ce qui est indispensable) : `chmod a+x leprogramme.pl` (a=all, x=eXecute). Si on désire que ses collègues puissent lire ses fichiers ou dossiers : `chmod -R g+r *` (`-R`=récursivement dans les sous-dossiers, g=group, r=read, w=write).
- Autres commandes d'usage courant : `cat, grep, chown, which, where, find, cut, uniq, sort`...
- De nombreux logiciels clickables peuvent aussi être lancés par le biais des commandes associées. Exemple : `mozilla`
  Note : quand un tel logiciel risque d'être utilisé longtemps, il est *conseillé* de le lancer en tâche de font (`&`) pour qu'il ne paralyse pas la fenêtre du terminal :
  ```
  mozilla &
  nedit file &
  ```
  Etc.

3.4 Empilages de commandes

C'est la force d'Unix/Linux : cet OS fonctionne sur le concept de brique élémentaire (logiciels et commandes élémentaires). Vous emboîtez l'une après l'autre pour obtenir le résultat de votre choix, que vous pouvez rediriger comme bon vous semble.

3.4.1 Emboîtements de commandes

Le trait d'union entre deux commandes s'appelle le *pipe*. C'est le symbole `|` . On l'obtient avec les touches Alt_Gr-6 sur un PC, Alt-Maj-L sur un Mac.

Voici deux exemples :

- `ls -l | less` : affiche page à page le contenu d'un dossier. Très utile s'il est volumineux.
- `cut -f2 file |sort |uniq -c |sort -rn` : compte les mots de la colonne 2 du fichier *file*, du plus fréquent au moins fréquent. Application : dénombrer les modalités d'un questionnaire sociologique. Détail des étapes :
  1. On sélectionne la 2^e colonne du fichier *file* (`cut -f2 file`).
  2. On trie le résultat (`sort`).
  3. On repère les lignes identiques (`uniq`) et on les compte (option `-c`). Le `sort` précédent était nécessaire car `uniq` ne fonctionne que sur des fichiers triés.
  4. On trie ce dernier résultat (`sort`) numériquement et non alphabétiquement (`-n`) du plus grand au plus petit (`-r` : reverse).

3.4.2 Usage du système de fichiers

Le résultat d'une commande (ou de plusieurs emboîtées) peut être redirigé vers des fichiers : `> file` (ou `>> file` si on veut éviter d'écraser le fichier *file* en cas de relance de la commande).

Exemple : `grep sub *pl | sort > mesroutines`
sélectionne toutes les lignes contenant le mot *sub* (qui définit une routine en Perl) dans les fichiers se terminant par *pl* (souvent des programmes Perl), les trie et copie le résultat dans le fichier *mesroutines*.

4 Unix dans Perl

Il est conseillé de connaître des rudiments de Perl avant de lire cette partie, qui donne très succinctement quelques moyens de manipuler *Unix/Linux* via *Perl*.

4.1 Commande `system`

On peut appeler une commande Unix (en fait, un programme externe) à partir d'un programme Perl : `system ("la commande que je désire");`

Cet appel se produit au moment demandé et le programme Perl se poursuit une fois la commande externe achevée.

Cas classiques : `wget, ps2gif`, etc. Il est conseillé d'appeler ces commandes via leur chemin d'accès complet : `/bin/rm` mieux que `rm` !

4.2 Variables

Il arrive que l'on désire glisser dans une variable le résultat d'une commande Unix. En ce cas, on utilise des apostrophes inverses :

```
$ladatedelinstant= ` date `;
```

4.3 Redirections

Il est utile de se rappeler la ligne

`open (L, ls |);`

Le fichier L contient donc le résultat de la commande `ls`

Application : travailler sur tous les fichiers d'un dossier donné. Détail :

```
open (L, "ls | ");
while (<L>)
{chop;
$fichier=$_;
print "On va travailler sur le fichier $fichier \n";
open (F, $fichier);
while (<F>)
{traitement du fichier}
close (F);
}
close (L);
```

Table des matières