ResEl 101 : Cours sur les outils Linux

Association ResEl

septembre 2005

Table des matières

1 Introduction

Ce cours a pour objectif de décrire les outils de base de Linux, qui sont très souvent utilisés, mais parfois dificilles à comprendre. Le chapitres décrivent les fonctionnalités les plus souvent utilisées de ces outils, d'un point de vue pratique, sans entrer dans le détail de leur fonctionnement ; le but est triple :

- donner une vue d'ensemble des possibilités offertes par les outils en ligne de commande ;
- une introduction aux outils plus simple que le man ;
- une utilisation comme un pense-bête auquel on peut souvent faire référence.

Il est supposé que vous connaissiez déjà les commandes de base comme `cd`, `ls`, `mkdir`, etc.

2 Fonctionnalités de base

2.1 sudo

sudo est un outil de gestion des droits : grâce à sudo on peut donner certains droits à un utilisateur, on peut par exemple lui donner les droits d'un autre utilisateur (appelé toto). Cet utilisateur n'aura qu'à taper `sudo -u toto commande` (`-u` comme user) pour exécuter *commande* en tant que toto. L'utilisateur aura alors à taper *son* mot de passe, et non celui de toto.

L'utilisateur peut même passer « dans la peau » d'un autre utilisateur (encore toto), en faisant `sudo su toto`. Il aura dés cet instant un shell en tant que toto. Pour sortir de ce sudo il suffit de taper `exit`.

Lorsqu'on ne fait pas référence à un utilisateur particulier, sudo va faire comme si c'était les droits superutilisateur qui étaient demandés. Le célèbre `sudo su` fait en effet passer en root ceux qui en ont le droit.

Certaines commandes nécessitant des droits plus hauts que ceux de l'utilisateur peuvement également être exécutées par lui. Par exemple on peut paramétrer sudo pour qu'un utilisateur puisse utiliser `halt` (commande nécessitant des droits superutilisateur) alors qu'il n'a pas les droits superutilisateur.

Le paramétrage de sudo peut être fait de façon très fine (cf. `man sudoers`), il se fait en éditant le fichier `/etc/sudoers` ; dans la distribution Debian, on ne peut pas éditer ce fichier directement, il faut passer par la commande `visudo`. Ce fichier est constitué de lignes de ce type (on ne s'intéresse qu'aux configurations simples) :

```
uid hosts=(utilisateur) commandes
```

- `uid` représente le login de l'utilisateur ;
- `hosts` les machines à partir de laquelle l'utilisateur peut utiliser sudo (voir la section sur ssh pour comprendre l'intérêt de ce paramètre), hosts est en général à ALL ;
- `utilisateur` est le nom d'utilisateur duquel l'utilisateur (celui défini par `uid`) peut prendre les droits. Lorsque ce paramètre est à ALL, l'utilisateur (uid) peut prendre les droits de tous les autres utilisateurs, y compris le superutilisateur ;
- `commandes` est la liste des commandes que l'utilisateur peut utiliser en prenant les droits d'autres utilisateurs.

Par exemple un `/etc/sudoers` contenant la ligne

```
riri riri.maisel=(fifi) moo

toto ALL=(ALL) ALL
```

va permettre :
- à l'utilisateur riri d'executer la commande `moo` avec les droits de fifi, en faisant `sudo -u fifi moo`, mais uniquement s'il se connecte depuis la machine riri.maisel ;
- à toto de faire ce qu'il veut.

2.2 Les archives

Les archives les plus souvent utilisées sont les archives tar, ce sont des fichiers qui contiennent plusieurs fichiers, ou même toute une arborescence de répertoires, le tout non-compressé.

Pour créer l'archive toto.tar contenant le fichier toto.fichier et le répertoire repertoto, il suffit de taper `tar cvf toto.tar toto.fichier repertoto` et pour extraire cette archive `tar xvf toto.tar` (c comme create, x comme extract).

Deux niveaux de compression sont ensuite possibles : gzip et bzip2. Le premier est rapide, le second très performant.

Pour compresser un fichier avec gzip, il suffit de faire `gzip lefichier`, on obtient alors `lefichier.gz`, que l'on peut décompresser avec un `gunzip lefichier.gz`.

bzip2 fonctionne de la même façon : `bzip2 lefichier` pour obtenir `lefichier.bz2`, et `bunzip2 lefichier.bz2` pour le décompresser.

Ces outils peuvent être combinés pour faire des archives compressées directement :

`tar czf toto.tgz toto.fichier repertoto` créé directement le .tar compressé, `.tar.gz` est souvent « abrégé » en `.tgz`. Il suffit de remplacer le c par le x dans les options de tar pour le décompresser.

`tar xjf toto.tar.bz2 toto.fichier repertoto` fait la même chose avec bzip2 (une fois encore le x sert à décompresser).

2.3 grep

`grep` est une commande extrêmement utile, beaucoup plus que ce que l'on pourrait croire au premier abord, vous serez très souvent amené à l'utiliser, c'est pourquoi elle apparaît dans ce poly.

La fonction de cette commande est de chercher un mot dans un ou plusieurs fichiers.

L'utilisation de base est `grep chaîne fichier`, où chaîne est la chaîne recherchée et fichier le fichier (texte) dans lequel on veut la chercher. `grep` renvoie alors la ou les lignes de `fichier` contenant *chaîne*.

On peut également rechercher une chaîne dans plusieurs fichiers, et même dans les dossiers, récursivement :

```
grep -R chaîne *
```

va renvoyer toutes les lignes contenant *chaîne* dans tous les fichiers et tous les sous-répertoires, en affichant les noms des fichiers dans lesquels la chaîne a été trouvée).

À l'inverse, si on ne veut voir aucune ligne contenant la chaîne donnée, on utilise `grep` avec le commutateur `-v` : `grep -v chaîne *`.

Mais l'utilisation la plus courante de `grep` est celle consistant à rediriger la sortie d'une commande vers grep pour n'avoir que les résultats intéressants. Par exemple si on veut avoir la liste des `.tex` dans un répertoire, il suffit de faire `ls | grep .tex`. En effet grep ne va séléctionner que les lignes contenant la chaîne *.tex* dans la sortie de `ls`.

2.4 ssh

`ssh` (Secured SHell) est un outil magique qui permet de se connecter sur une machine distante. Il faut pour cela que la machine distante aie un sshd installé (disponible uniquement sur les systèmes GNU). Cet outil permet d'avoir un shell sur une machine sur laquelle on a un compte. La commande de base est la suivante :

```
ssh toto@titi.maisel.enst-bretagne.fr
```

`titi.maisel.enst-bretagne.fr` est ici le nom de la machine sur laquelle on veut se connecter, et `toto` le login d'un utilisateur qui a un compte sur la machine. Le mot de passe de `toto` est alors demandé. On obtient alors un prompt comme si on était sur la machine `titi`, en ligne de commande.

Il est également possible de lancer des programmes en mode graphique : la sortie graphique des programmes que l'on lancera sera redirigée vers la sortie graphique de la machine sur laquelle on est. Il faut pour cela activer l'option `-X` dans ssh, ce qui donnera une ligne comme `ssh -X toto@titi.maisel.enst-bretagne.fr`.

Pour terminer une connexion ssh, il suffit de taper `exit`.

2.5 scp

Pour faire des transferts de fichiers d'une machine à une autre simplement, ce n'est en général pas simple (installer un serveur ftp n'est pas immédiat), surtout lorsque ces fichiers sont critiques. Heureusement la commande scp est là ! Cette commande permet le transfert de fichiers entre des machines sur lesquelles vous avez un compte.

Son utilisation de base est très simple :

```
scp fichier1 fichier2 fichier3 toto@titi.maisel.enst-bretagne.fr:repertoire
```

va copier les fichiers `fichier1`, `fichier2` et `fichier3`, sur la machine titi dans le répertoire `repertoire`.

`repertoire` peut être soit :
- le chemin absolu d'un répertoire, par exemple `/usr/src/linux` ;
- $\sim$, ce qui mènera vers le répertoire de l'utilisateur, ici `/home/toto` ;
- le chemin relatif d'un répertoire, par exemple `public_html`, qui mènera vers `/home/toto/public_html`.

3 Installation et compilation

3.1 L'outil apt

L'installation en ligne de commande se fait très simplement grâce à l'outil `apt` : en effet, il permet de rechercher, d'avoir la description et d'installer les paquets.

Pour rechercher un paquet à partir d'un mot clé, il suffit de faire `apt-cache search monpaquet` où *monpaquet* est une partie du nom du paquet recherché ou un mot clé le caractérisant.

Lorsque l'on n'est pas certain du paquet que l'on veut installer, on peut avoir la description d'un paquet grâce à la commande `apt-cache show monpaquet`.

Pour installer un paquet, il faut utiliser `apt-get install monpaquet`, où *monpaquet* est le nom exact du paquet que l'on veut installer.

Les mises à jour du système se font en deux étapes : `apt-get update` met à jour la base de données, et `apt-get upgrade` met à jour les paquets dont la version est plus ancienne que leur entrée dans la base de données.

Il est important de configurer les sources sur lesquelles `apt` va chercher les paquets, afin de ne pas trop prendre de bande passante inutilement sur le net, il faut pour cela modifier le fichier `/etc/apt/sources.list` (c'est le même fichier que pour configurer les sources de synaptics). La page `http://miroir.maisel.enst-bretagne.fr/Bienvenue.php` explique comment bien faire ces changements.

3.2 Compilation à partir des sources

La compilation à partir des sources est quelquechose à savoir faire car c'est très souvent utile, et ce n'est pas aussi monstrueux que ça en a l'air.

Les sources d'un paquetage se présentent souvent sous la forme d'un fichier `.tar.gz`. On peut trouver ce fichier sur Internet ou on peut l'obtenir par la commande `apt-get source monpaquet`, qui va télécharger les sources du paquet actuel de la distribution sous laquelle vous êtes. La plupart du temps, les sources obtenues avec apt contiennent les sources du paquet standard plus un patch spécifique à la distrib, qu'il faut appliquer. partie sur les patch et les diff

Un problème de la compilation est la gestion des dépendances. En effet pour être compilés, certains paquetages nécessitent les sources de certaines librairies, et nécessitent également d'autres paquetages pour fonctionner. Les dépendances peuvent être résolues automatiquement par apt, grâce à la commande `apt-get buid-dep monpaquet`.

La première étape pour compiler est de se rendre dans le repertoire et de taper `./configure`. Cela appelle un script qui vérifie si le paquet peut compiler, si les dépendances son bonnes, etc. Ce script peut avoir plusieurs options qui dépendent des paquets, mais la plus importante est l'option `-prefix`, puisqu'elle permet de paramétrer dans quel répertoire le fichier exécutable va être mis. Il faut le mettre dans `/usr/bin` la plupart du temps, il faut donc taper `./configure -prefix=/usr` pour configurer un paquet correctement (le `/bin` est ajouté automatiquement).

Une fois que `./configure` ne rapporte plus d'erreur, on peut passer à l'étape de compilation proprement dite en tapant simplement `make`.

La troisième étape doit se faire avec les droits superutilisateur (les deux autres non) : `sudo make install` va mettre les exécutables dans les bons répertoires et va finir l'installation.

4 Appendice A : tutoriel d'introduction à Vim

4.1 Présentation

Cette partie est une Introduction à Vim (Vi iMproved, le successeur de Vi), un éditeur de texte utilisable en mode graphique, mais surtout très simple à utiliser en mode texte, ce qui en fait un bon outil pour l'édition de fichiers sur des serveurs distants, lorsque ceux-ci n'ont pas de serveurs graphiques.

Cette introduction présente les fonctionalités de Vim les plus utilisées, dans l'ordre croissant de difficulté.

4.2 Les commandes de base

4.2.1 Ouvrir et créer un fichier

Commençons donc par le commencement : comment ouvrir un fichier ?

Eh bien c'est très simple, il suffit de taper dans son shell préféré `vim fichier` où *fichier* est un nom de fichier.

Sur les distributions Linux récentes, Vi est remplacé par Vim, ce qui fait que lorsque l'on tape `vi`, c'est en fait Vim qui se lance (c'est un alias). On tape donc la plupart du temps `vi fichier`, mais il reste des dinosaures sur lesquels ce n'est pas le cas, méfiance donc.

Si vous voulez créer un fichier c'est exactement pareil : `vi toto` (ou `vim toto` sur les vieux OS) vous ouvrira un fichier vide et dès que vous l'enregistrerez il s'appellera `toto`.

4.2.2 Les différents modes de Vim

Maintenant que vous avez ouvert un fichier, vous avez certainement remarqué que lorsque vous tapez sur les lettre de votre clavier, cela donne des résultats étranges ou même des "biiip". Cela est dû au fait que Vim peut tourner sous plusieurs modes, et vous êtes dans le mode pour entrer des commandes, nous allons donc arranger cela.

Vim possède en fait (pour simplifier) trois modes : le mode *insertion*, qui permet d'insérer des caractères dans un texte (le mode "normal" si on veut), le mode *remplacement*, qui fait sensiblement la même chose mais en remplaçant les caractères, et enfin le mode *commande* qui permet d'effectuer toutes les commandes.

Pour passer au mode *commande* il suffit d'appuyer sur `Echap`. La touche `Inser` permet de passer au mode *insertion* depuis le mode `commande`, puis de switcher entre le mode *insertion* et le mode *remplacement*. Pour taper du texte quand vous venez d'arriver dans Vim il faut donc appuyer sur `Inser`.

4.2.3 Quitter et enregistrer

Vous savez désormais taper du texte, mais il va falloir l'enregistrer, quitter, etc. Pour cela il faut être en mode *commande* et taper une de ces commandes (il y a presque toujours des `:` avant les commandes, il ne faut pas les oublier)

- `:q` pour quitter ;

– `:w` pour enregistrer (vous pouvez enregistrer dans un autre fichier en tapant `:w le nom du fichier`;
– `:x` (ou `:wq`) pour les deux à la fois.

Lorsque vous voulez quitter sans enregistrer (vous ne pouvez pas par défaut si vous avez fait des modifications), il faut forcer la sortie, pour cela rajoutez un `!` à la fin de votre commande : (pour forcer la sortie par exemple faites `:q !`, `:w !` sert également souvent).

4.2.4 La coloration syntaxique

La coloration syntaxique est un outil extrêmement pratique puisqu'il permet de se repérer très vite dans un code, de quel type qu'il soit (Vim reconnaît plus d'une centaine de syntaxes). Vous verrez que très rapidement vous ne pourrez plus vous en passer !

La commande pour ce truc magique est `:syn on` (*syn* ou *syntax*). Si vraiment enfin vous avez mal au crâne avec toutes ces couleurs, un petit `:syn off` et tout redevient gris.

Une commande qui peut parfois servir sur les portables (ou quand les gens qui ont des portables mettent ça par défaut et qu'il faut l'enlever) : pour régler la luminosité (ou plutôt l'épaisseur) du texte, il suffit de taper `:set bg=dark` (ou `:set bg=light` pour l'autre sens).

4.3 Les copier-coller

4.3.1 Le mode visuel

Pour faire des copier-coller, une première étape est de pouvoir séléctionner un morceau de texte. Pour cela vous avez deux modes visuels : le mode *visual* et le mode *visual_line*. Vous pouvez y accéder en mode *command* en appuyant simplement sur `v` pour le mode *visual* et sur `V` pour le mode *visual_line*. Une fois dans un de ces modes vous pouvez déplacer le curseur pour sélectionner des caractère ou des lignes.

Pour sortir de ce mode appuyez simplement sur `Echap`.

4.3.2 Copier-coller

Pour copier ce que vous avez sélectionné il suffit de taper sur `y`, pour couper `d` et pour coller `p` (`P` pour coller avant le curseur).

Notez que vous pouvez directement copier (ou couper) une ligne sans la sélectionner en appyant deux fois sur `y` (ou `d`).

Enfin, quelque chose de très pratique est le copier-coller à la souris : sélectionnez quelque chose avec votre souris (enfin sous linux hein...), ça copie, puis sous Vim mettez-vous en mode *insertion*, placez le curseur là où vous voulez insérer ce que vous avez copié et faites un coller avec votre souris (bouton du milieu, ou bouton droite + bouton gauche).

Par contre cette méthode fait souvent n'importe quoi au niveau de l'indentation. Pour régler ce problème regardez la partie sur l'indentation.

4.3.3 Gérer plusieurs fichiers

Bon, tout ça c'est bien beau, mais comment on fait pour faire des copier-coller d'un fichier à l'autre ? Parce que ça n'a pas l'air super pratique ce mode texte là...

Eh bien là encore tout est prévu : vous pouvez ouvrir plusieurs fichiers dans le même écran, en séparant votre écran en 2 (ou plus), il suffit pour cela de taper `:sp fichier` (*sp* ou *split*) où *fichier* est le chemin relatif du fichier que vous voulez ouvrir par rapport au répertoire courant. Par exemple si vous voulez ouvrir un autre fichier qui est dans le même répertoire, donnez juste le nom du fichier, pour un fichier qui est dans le répertoire père, tapez `:sp ../fichier`, etc.

Pour ceux qui ne se rappellent plus bien quel fichier ils veulent ouvrir, un `:E` ouvre un petit menu ou vous pouvez vous balader dans les répertoires pour rechercher le fichier à ouvrir (très pratique).

Pour passer d'un fichier à l'autre, faites (en mode *commande*, est-il besoin de le préciser ?) un `C-w` puis une direction (haut ou bas), et vous allez éditer le fichier situé dans cette direction sur votre écran. Vous pourrez traiter ce fichier totalement indépendemment de l'autre (si vous faites un `:q` seul celui sur lequel vous êtes sera fermé, etc.).

4.4 Quelques astuces très pratiques

4.4.1 Le fichier `.vimrc`

Si vous voulez sauvegarder votre configuration de Vim (par exemple pour mettre la coloration syntaxique par défaut). Ce fichier est contenu dans le dossier home de chaque utilisateur (donc la configuration peut être différente en fonction de l'utilisateur), mais il n'est pas forcément présent. Dans ce fichier vous pouvez mettre tout ce que vous taperiez dans Vim : par exemple pour mettre la coloration syntaxique par défaut, vous pouvez faire un fichier `.vimrc` contenant la ligne

```
:syn enable
```

4.4.2 Recherche et substitution

Pour rechercher une chaîne dans un texte rien de plus simple il suffit de taper `/chaîne` où *chaîne* est la chaîne que vous voulez rechercher dans le texte. Pour aller au résultat suivant, faites `n` (comme *next*) et le résultat précédent `N`.

Pour substituer une chaîne par une autre, il y a énormément de possibilités qui ne seront pas détaillées ici, mais voici tout de même la méthode pour remplacer une chaîne par une autre : pour le faire dans tout le document faites `:%s/chaîne1/chaîne2/g` ; si vous enlevez le `g` de la fin, vous ne substituerez que la première occurence de chaque ligne de la *chaîne1* par la *chaîne2*.

Faites attention aux caractères d'échappement : faites un plutôt qu'un `/` (qui serait interprété) dans vos chaînes. Et il fait faire la distinction entre vos chaînes qui commencent par `/` et celles qui ne commencent pas par `/` (en xhtml par exemple).

Pour plus d'infos sur les énormes possibilités de Vim, vous pouvez regarder l'URL documentant les regexp Vim indiquée dans la section Url.

Enfin quelque chose d'extrêmement pratique pour le debug de programme est d'aller directement à une ligne. Pour cela il suffit de taper `:x` où x est le numéro de la ligne à laquelle vous voulez aller.

4.4.3 Undo - redo

S'il y a bien quelque chose de pratique c'est d'annuler la boulette que l'on vient de faire. C'est très simple : il vous suffit de faire `u` (comme *undo*) pour annuler (on peut remonter assez loin tant qu'on n'a pas fermé le fichier). Pour annuler son annulation la commande est `C-r`.

Une fonction pratique également est celle consistant à répéter la dernière action, pour cela faites `.` et le tour est joué.

Pour rappeler une commande que vous avez tapé, il vous suffit de taper `:` puis de naviguer avec la flèche du haut et celle du bas entre les commandes, comme dans un shell.

4.4.4 L'aide

Dans Vim pour connaître toutes les options et possibilités d'une commande rien de plus simple tapez `:h commande` (*h* ou *help*) où *commande* est la commande sur laquelle vous voulez vous documenter.

4.5 Aller plus loin

4.5.1 Les formats de fichier

Il y a plusieurs types de formats de fichier texte, surtout dépendant des OS (c'est pour ça que par exemple dans notepad vous voyez tout un fichier qui ne fait qu'une seule ligne quand il vient de Linux). Et c'est pas super la classe d'avoir un script en format Windows sur un serveur Linux. Vim peut convertir des fichiers d'un type à l'autre, simplement en tapant `:set ff=truc`, où *truc* est *dos*, *unix* ou *mac*. Vim ouvre automatiquement les fichiers avec le bon type.

4.5.2 Éxecuter des commandes

Pour éxecuter des commandes shell il suffit la plupart du temps de faire `:!commande` où *commande* est ce que vous auriez tapé dans un terminal. Une fois la commande terminée, vous pourrez appuyer sur entrée pour revenir à Vim.

Pour avoir carrément un shell vous pouvez taper `:sh`. Pour revenir à Vim, faites un `C-d`.

4.5.3 L'encodage des caractères

Vim peut ouvrir et encoder des fichiers dans plusieurs charsets. Pour changer il suffit de taper `:set encoding=truc` où *truc* peut être *utf8* ou *latin1* couramment.

Pour afficher correctement un fichier encodé dans un charset qui ne convient pas (par exemple si vous voyez des choses bizarres du style *Ä©*), il faut redéfinir l'encodage d'affichage en faisant un `:set termencoding=truc` avec les mêmes possibilités qu'avant.

4.5.4 La taille des lignes

Quelque chose d'assez énervant dans Vim est de devoir parcourir caractère par caractère une longue ligne pour arriver à un endroit précis, et l'autre problème est que quand vous envoyez ça par mail qui doivent être formatés à 80 caractères par ligne.

Le mieux est alors de formater le text bien en tapant `gqgq` (sans les `:` habituels, donc vous ne verrez pas `gqgq` à l'écran, mais ça marche quand même) lorsque vous avez sélectionné du texte.

Vous pouvez également dire à Vim de formater automatiquement les lignes à 80 (ou 72 par exemple) caractères en tapant `:set textwidth=80` en début d'édition. Vous pouvez aussi pourquoi pas le mettre dans votre `.vimrc` .

4.5.5 Les problèmes d'indentation

Lorsque vous faites un copier-coller à la souris il est fort probable que l'indentation soit foireuse. Pour supprimer ce problème, faites un `:set noautoindent` avant de copier avec la souris, et un `:set autoindent` après (pour retrouver l'autoindentation, qui est tout de même pratique).

L'autre solution, moins élégante consiste à sélectionner les lignes en mode visuel et à taper `=`, cela supprimera complètement l'indentation. Pour faire ça sur tout le fichier faites `gg=G` (les `g` et `G` sont pour la sélection, leur fonctionnement n'est pas détaillé ici).

4.6 URL intéressantes

- `http://www.geocities.com/volontir/`pour les regexp sous Vim ;
- `http://vimdoc.sourceforge.net/htmldoc/mbyte.html` : la documentation de Vim (ce que vous avez en tapant `:h`) ;
- `http://www.vi-improved.org/tutorial.php` pour apprendre à Socrate à utiliser Vim : tutoriel simple mais complet, en anglais ;
- `http://tnerual.eriogerg.free.fr/vimqrc-fr.pdf` : un tableau des commandes de Vim, très utile ;
- `http://en.wikipedia.org/wiki/Editor\_war` : une comparaison objective de Vim et Emacs.

5 Appendice B : tutoriel d'introduction à Screen

5.1 Présentation de Screen

Screen est un outil magique pour tout le monde et encore plus pour le travail en commun, sur des serveurs par exemple. Il permet de lancer des consoles (avec des commandes actives dedans) sur une machine, de les laisser sur la machine et de pouvoir les récupérer plus tard, même à distance. Plusieurs personnes peuvent également être sur une même console en screen, ce qui est pratique pour faire des démonstrations sur des serveurs distants par exemple. Bref, l'essayer c'est l'adopter.

5.2 Les sessions

Lorsque l'on utilise screen, on utilise en premier des sessions. Une session, pour l'instant est une console. Cette console est similaire à une console ordinaire, et peut être détachée, c'est à dire "laissée" sur la machine, pour être récupérée plus tard.

Pour lancer une session de screen, rien de plus simple : tapez simplement `screen` dans votre shell préféré. La seule chose à vérifier est l'utilisateur avec lequel vous lancez le screen : en effet une session screen root ne pourra pas être récupérée par des utilisateurs non-root (et attention, le sudo pose des problèmes dans ce cas).

Une fois que vous êtes dans cette session, vous aurez l'impresssion d'être dans une console tout à fait normale. Pour la détacher, faites simplement `C-a C-d`. Votre session est détachée, vous pourrez la récupérer plus tard.

Vous verrez que `C-a` est la combinaison de touche pour dialoguer avec le screen.

Pour récupérer une session, il suffit de faire `screen -r`, et le tour est joué (en faisant bien attention à l'utilisateur).

Dans le cas où plusieurs personnes veulent accéder à la même session, il faut pouvoir entrer dans une session qui n'est pas détachée, pour cela les utilisateurs devront taper `screen -x` au lieu de `screen -r`. Ce fonctionnement peut être détourné pour faire office de messagerie instantanée entre des personnes sur une même machine ;)

5.3 Avoir plusieurs sessions

Le problème lorsque vous utilisez `screen -r` est quand vous voulez faire plusieurs sessions, en effet, avec seulement `screen -r`, on ne peut pas décider quelle session reprendre lorsqu'on en a plusieurs. Pour cela vous avez `screen -ls`, qui liste les sessions actives, dans un ordre qui semble être l'ordre chronologique de leur création (vous obtenez le même résultat avec un `screen -r` lorsque vous avez plusieurs sessions). Dans cette liste vous pouvez voir le pid des sessions de screen. Pour revenir à une session particulière il faut faire `screen -r x`, où x est le pid de la session en question.

Une autre solution infiniment plus élégante consiste à donner un nom à une session, pour cela lorsque vous la lancez faites `screen -S nom`, où *nom* sera le nom de la session. Pour la récupérer faites simplement `screen -r nom`.

5.4 À l'intérieur d'une session

Il a été dit au début qu'une session était une console, mais en fait une session de screen peut contenir jusqu'à 10 fenêtres ! À l'intérieur d'une session, pour créer une fenêtre il suffit de faire `C-a C-c` (*c* comme *create*). La navigation entre les fenêtres peut se faire grâce à `C-a n` (comme *next*) et `C-a p` (comme *previous*) ; ou plus simplement `C-a "` pour avoir une liste des fenêtres et de leur action et choisir. La multiplicité des fenêtres est plus souvent utilisée que celle des sessions, étant plus simple.

Pour avoir une liste complète des nombreuses commandes possibles pour les fenêtres, regardez la section *DEFAULT KEY BINDINGS* du manuel de screen (`man screen`).

Dernière astuce : l'option `-U` qui permet d'avoir un screen en UTF-8.