



Initiation à Unix

L'environnement de travail à l'Ensimag

Olivier Alphan Nicolas Berthier Bernard Cassagne
Catherine Cassagne Giovanni Funchal Grégory Mounié
Matthieu Moy Simon Nieuviarts Florence Perronnin

Rentrée 2017

Table des matières

1	Introduction	7
1.1	Bienvenue	7
1.2	Version PDF de ce document	7
1.3	Au sujet du stage	7
1.4	EnsiWiki	7
1.5	Jeu de piste	8
1.6	Examen de TP	8
1.7	Notations utilisées dans ce document	8
2	Premiers contacts avec l’environnement	11
2.1	La connexion à un ordinateur	11
2.1.1	Démarrage d’un PC	11
2.1.2	La procédure de login	11
2.1.3	La session	12
2.1.4	La procédure de <i>Verrouillage</i>	12
2.1.5	La procédure de <i>logout</i>	12
2.2	L’environnement graphique	13
2.2.1	L’environnement graphique : Gnome 3 Classic	13
2.2.2	Quelques exercices avec le clavier Azerty	14
2.3	Illustration de quelques différences entre Unix et Windows	14
2.3.1	La souris a trois boutons !	14
2.3.2	Des astuces qui changent la vie	15
2.3.3	Les bureaux virtuels	16
2.3.4	Le terminal et le shell	16
2.4	Règles élémentaires de sécurité	18
2.4.1	Les risques	19
2.4.2	Un vrai mot de passe	19
2.5	L’intranet de l’Ensimag	21
2.6	Un problème ?	22
3	Internet et courrier électronique à l’Ensimag	23
3.1	Navigation sur le Web : Firefox	23
3.2	Votre adresse de courriel	23
3.3	Netiquette	24
3.3.1	Rédaction	24
3.3.2	Transfert, envoi à de multiples personnes	25
3.3.3	Pièces jointes	25
3.4	Un client de messagerie simple à utiliser : Thunderbird	26

3.5	Lire et envoyer des mails depuis n'importe où : le <i>webmail</i>	26
3.6	IMAP	27
3.6.1	Sauvegarder ses courriels dans des dossiers	28
3.6.2	L'envoi est différent de la lecture	28
3.7	Les lecteurs de courriel textuels : Alpine, Mutt	28
4	Le b.a.-ba pour survivre sous Unix	31
4.1	Système d'exploitation	31
4.2	Commandes et interpréteur de commandes	31
4.2.1	Notion de commande	31
4.2.2	Notion d'interpréteur de commande	32
4.3	Le système de fichiers d'Unix	33
4.3.1	Notions de fichier et de répertoire	33
4.3.2	Un gestionnaire de fichiers graphique : <i>nautilus</i>	34
4.3.3	Définitions	34
4.3.4	Naviguer dans l'arborescence	37
4.3.5	Déplacer, copier, supprimer des fichiers	37
4.3.6	L'arborescence de répertoires usuelle sous Unix	37
4.3.7	Manipulations de base sur les fichiers et répertoires	39
5	Applications utiles	43
5.1	Quelques applications déjà vues par ailleurs	43
5.2	Bureautique : <i>LibreOffice</i>	43
5.3	Visualiser des images : <i>Eye of Gnome</i> , <i>Gthumb</i>	44
5.4	Visualiser des documents PDF : <i>Evince</i> , <i>Acrobat Reader</i>	44
5.5	Dessin et retouche d'images : <i>GIMP</i>	44
5.6	Composition de documents : <i>L^AT_EX</i>	44
5.7	Tout ça est gratuit ?	45
6	Les éditeurs de texte	47
6.1	<i>Atom</i>	47
6.2	<i>GNU Emacs</i>	48
6.3	L'éditeur <i>Vim</i>	48
7	Commandes et outils indispensables	51
7.1	Les informations associées aux fichiers et répertoires	51
7.1.1	<i>ls</i> revisité	51
7.1.2	Notion de répertoire courant : <i>cd</i> et <i>pwd</i>	53
7.1.3	Quelques répertoires particuliers	53
7.1.4	Droits des fichiers et répertoires	54
7.1.5	Modification des droits d'accès : <i>chmod</i>	55
7.1.6	Utilisation de l'espace disque : <i>du</i> et <i>df</i>	56
7.2	Manipulation de fichiers : contenu, organisation	56
7.2.1	Visualisation du contenu : commandes <i>more</i> et <i>less</i>	56
7.2.2	Astuce : Utilisation efficace du terminal	57
7.2.3	Notion de lien symbolique	58
7.2.4	Recherche de fichier : <i>find</i> et <i>grep</i>	58
7.3	Compression et archivage	60
7.3.1	Compression avec <i>gzip</i>	60

7.3.2	Archivage avec <code>tar</code>	61
7.3.3	Archivage puis compression avec <code>tar</code>	61
7.3.4	Archivage et compression avec <code>zip</code>	62
7.4	Impression de documents : <code>a2ps</code> , <code>cancel</code> et <code>lpstat</code>	62
7.4.1	Imprimer un document	63
7.4.2	Plus d'information	64
7.5	Obtenir de l'aide : <code>--help</code> et <code>man</code> et <code>info</code>	64
7.6	Que se passe-t-il sur le serveur ?	65
7.6.1	Notion de programme	65
7.6.2	Notion de processus	65
7.6.3	La différence entre un processus et une fenêtre	66
7.7	Arrêter des processus, plus ou moins gentiment	67
7.8	Redirection des Entrées Sorties	68
7.9	Les tuyaux, ou « pipe »	70
7.10	Outils spécifiques à l'Ensimag	71
8	Utilisation du <i>shell Bash</i>	73
8.1	Les 1001 manières d'entrer une commande	73
8.1.1	La complétion des noms	73
8.1.2	Gestion de l'historique des commandes	74
8.1.3	La convention tilde	74
8.2	Parler de plusieurs fichiers à la fois : les <i>wildcards</i>	75
8.3	Lancer plusieurs commandes en même temps	76
8.3.1	Première solution : <code>Control-z</code> , <code>bg</code>	76
8.3.2	Deuxième solution : lancer en tâche de fond avec « <code>&</code> »	76
8.4	Configuration de Bash : le fichier <code>.bashrc</code>	77
8.4.1	Un fichier lu à chaque lancement	77
8.4.2	Les variables	77
8.4.3	Alias	78
8.5	Où sont rangées les commandes ?	79
8.6	Exemples de configuration de <code>bash</code>	80
8.6.1	Des alias pour le confort	80
8.6.2	Des alias pour éviter les erreurs	81
8.6.3	La complétion programmable	81
9	Clients légers, poste de travail virtualisé, et serveurs	83
10	Accéder à l'Ensimag depuis l'extérieur et les ordinateurs portables	85
10.1	Les ordinateurs portables personnels à l'Ensimag	85
10.2	Les mails : cf. <i>Webmail</i>	85
10.3	Transférer des fichiers : <code>sftp</code> et ses amis	85
10.4	Accéder directement au serveur Unix : SSH	86
10.5	Accéder à Unix depuis Windows et vice-versa	88
10.5.1	Utiliser Windows à l'Ensimag	88
10.5.2	Autres solutions pour accéder à ses fichiers, exécuter des commandes à distance	90
10.5.3	Plus d'information	90
10.5.4	Les différences de codage des caractères	90
10.6	Et maintenant ?	92

10.6.1	Installer Linux sur un ordinateur personnel	92
10.6.2	Continuer à apprendre	93
Index		94

Chapitre 1

Introduction

1.1 Bienvenue

Tout d'abord :

Bonjour, et bienvenue à l'Ensimag !

1.2 Version PDF de ce document

Il existe aussi une version de ce document dans le format HTML, optimisée pour la lecture à l'écran. Voir la page chamilo du cours.

1.3 Au sujet du stage

Le stage que vous suivez ici a pour but de vous familiariser avec l'environnement de travail à l'Ensimag.

Cette brochure offre un certain nombre d'explications et des suggestions de manipulations. Nous vous demandons de lire la brochure et de réaliser les manipulations. Si vous ne réussissez pas ou si vous ne comprenez pas quelque chose, ne passez pas outre, demandez de l'aide à la personne qui vous encadre (en direct, ou bien par e-mail en dehors des séances si besoin). À la fin de ce stage, vous saurez vous connecter à un ordinateur via le réseau, créer et gérer des fichiers, créer et exécuter des programmes.

Ce document se veut d'abord pédagogique. Il fait donc peu d'hypothèses sur vos connaissances préalables de l'informatique ou d'Unix. Certaines parties peuvent sembler presque triviales si vous savez déjà vous servir d'un clavier et d'une souris. Néanmoins le but étant aussi d'attiser votre curiosité, chaque partie contient des aspects plus avancés.

1.4 EnsiWiki

En parallèle avec cette brochure papier, un Wiki est à votre disposition : EnsiWiki. Il contient des informations un peu plus techniques que ce document, et il devrait contenir les réponses aux questions que vous vous poserez pendant votre scolarité à l'Ensimag.

Vous trouverez sa page d'accueil à l'adresse suivante :

<https://ensiwiki.ensimag.fr/>

Cette partie de l'intranet est un Wiki, c'est-à-dire qu'il est éditable directement depuis votre navigateur web. Pour éditer une page, c'est très simple : il suffit de se connecter (lien en haut de chaque page) avec votre identifiant Ensimag habituel, puis de cliquer sur l'onglet **éditer** de la page de votre choix. Tous les étudiants ont les droits en écriture, vos contributions sont les bienvenues, de la simple correction de faute d'orthographe à la création d'un article complet. En fait, une bonne partie du contenu actuel du Wiki a été créé par des étudiants !

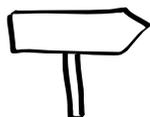
Nous vous invitons en particulier à lire la FAQ (*Frequently Asked Questions*, ou « Foire Aux Questions » dans la langue de Molière), accessible dans la barre latérale du Wiki.

D'une manière générale, vous trouverez beaucoup d'informations sur l'intranet de l'école :

<https://intranet.ensimag.fr/>

1.5 Jeu de piste

En plus des exercices proposés dans ce document, vos enseignants vous ont préparé un jeu de piste à travers les systèmes informatiques de l'école. La première étape du jeu de piste se trouve sur EnsiWiki, dans la page « Stage Unix de Rentrée ». Cette étape donne les instructions pour arriver à la seconde étape, qui donne les instructions pour la troisième, et ainsi de suite. Au fur et à mesure que vous avancerez dans ce document, vous devriez pouvoir avancer dans les étapes du jeu de piste. Certaines questions sont difficiles, mais toutes sont réalisables avec un peu de persévérance et l'aide de vos enseignants. Pour vous guider dans votre progression, vous trouverez dans ce document des remarques vous disant jusqu'où vous êtes censés être arrivés, comme ceci :



Jeu de piste

Pour l'instant, vous n'avez pas encore commencé, c'est juste un exemple !

1.6 Examen de TP

Pendant vos partiels de mi-semestre, un examen de TP permettra de vérifier que vous avez assimilé les bases suite à ce stage de rentrée. L'examen est constitué d'une série de questions fortement inspirées du jeu de piste. Pour chaque question, une manipulation sur machine (en général en ligne de commande) vous permettra d'obtenir la réponse à cette question. L'examen comportera beaucoup de questions, chacune étant relativement simple si vous avez terminé et compris le jeu de piste, et lu ce document en entier.

1.7 Notations utilisées dans ce document

Quand on parle d'un langage, on se pose le problème de déterminer ce qui est langage et ce qui est méta-langage. En d'autres termes, quand on dit : écrivez votre nom, faut-il

écrire « votre nom » ou faut-il écrire « Pierre Durand » ? Nous leverons cette ambiguïté à l'aide des notations suivantes :

1. ce qui est écrit avec une fonte à largeur constante **comme ceci** fait partie du langage ;
2. ce qui est écrit en italique *comme ceci* est du méta-langage.

Quand on dit : taper *Entrée*, cela ne signifie pas taper E,n,t,r,é,e mais taper sur la touche marquée Entrée (Elle sera notée `entrée` dans la suite). Quand on dit : taper *rm nom-de-fichier*, **rm** doit être tapé tel quel, mais *nom-de-fichier* doit être remplacé par un nom de votre choix.

Le document est ponctué par des remarques, des petites manipulations à effectuer, des avertissements sur les points les plus importants. Les notations suivantes sont utilisées.



Le saviez-vous ?

Exemple de remarque pour utilisateurs avancés

Ces remarques ne sont pas indispensables, et les débutants peuvent passer dessus rapidement.

Exercice 1 (Exemple d'exercice) *Voici un exemple d'exercice. Ces exercices sont des petites manipulations que vous pouvez faire au fur et à mesure de votre lecture.*



Attention !

Exemple de point délicat

Voici un exemple de point délicat à lire avec beaucoup d'attention.

Chapitre 2

Premiers contacts avec l'environnement

2.1 La connexion à un ordinateur

Le but de ce chapitre est de faire connaissance avec les PC Linux et d'apprendre la procédure de connexion à un ordinateur.

2.1.1 Démarrage d'un PC

Première chose à faire, bien sûr : allumer la machine ! Les PC de l'école peuvent exécuter plusieurs systèmes d'exploitations, c'est à l'utilisateur de choisir lequel il veut utiliser au démarrage. Après quelques secondes, un menu s'affiche, sélectionnez (avec les flèches haut/bas du clavier) **Linux CentOS 7**.

2.1.2 La procédure de login

La machine choisie vous invite à renseigner deux champs : **login** et **password**. Le *login* est un nom qui vous identifie ; on choisit en général le nom de famille, dans votre cas suivi de l'initiale du prénom. Il fait en général moins de 8 caractères, pour des raisons pratiques. Le *password* est un mot de passe, qui contribue à garantir que vous serez seul à pouvoir accéder à vos données. Pour la première connexion, on vous indique votre *login* (définitif) et un *password* provisoire.

Exercice 2 (Procédure de login) *Un identifiant (login) et un mot de passe ont dû vous être distribués. Entrez-les et validez avec la touche . Les majuscules et les minuscules sont différenciées. Vous n'aurez aucun mal à savoir si vous avez réussi ou pas. En cas d'échec, vous avez pu vous tromper dans le login, dans le password ou dans les deux. Essayez encore.*

Pour éviter qu'un individu mal intentionné puisse connaître votre mot de passe ou même le nombre de lettres qui le constitue en regardant votre écran, il est fréquent sous Unix que rien ne s'affiche pendant que vous entrez votre mot de passe. Si c'est le cas, pas de panique : c'est normal !

2.1.3 La session

Après la procédure de *login*, une *session* s'ouvre et vous avez maintenant accès à vos données, à votre espace de travail et à vos préférences (profil). Cette session vous est personnelle : elle n'interfère pas avec celles des autres utilisateurs connectés.

2.1.4 La procédure de *Verrouillage*

Quand vous laissez votre poste de travail pour une courte pause, il est indispensable que votre session soit verrouillée afin d'éviter qu'elle soit utilisée par un tiers. Attention toutefois de ne pas abuser du verrouillage de session, utilisez-le uniquement pour de courtes pauses et si vous êtes sûr de revenir sur le poste dans un délai relativement court. Sinon, déconnectez-vous pour libérer votre machine.

Exercice 3 (Verrouillage) *Votre session étant ouverte, verrouillez votre session (menu  en haut à droite de l'écran). Une fois votre session verrouillée, appuyez sur une touche. Vous devez alors vous identifier pour retrouver votre environnement de travail tel que vous l'avez laissé.*

2.1.5 La procédure de *logout*

Selon toute logique, une session devrait se terminer lorsque l'utilisateur arrête de se servir du système. Pour cela, vous devez demander explicitement l'arrêt de la session (telle une déconnexion).

Oublier de se déconnecter n'est pas sans poser des problèmes, notamment en matière de sécurité. Que peut-il se passer si vous quittez votre place en laissant une connexion active ? Une personne arrivant après vous aurait la possibilité d'accès à vos données ; les machines ne feront pas la distinction ! Elle pourrait aussi agir en votre nom.



Attention !

Gestion de votre identité

Prenez donc l'habitude de ne partir qu'après avoir terminé proprement votre session (ou éventuellement verrouillé votre écran), et conservez votre mot de passe aussi secret que votre code de carte bancaire ! Nous verrons un peu plus loin comment choisir un mot de passe fiable.

Avant de vous déconnecter, prenez l'habitude de fermer proprement toutes vos applications : cela évitera de laisser des applications tourner et consommer des ressources sur l'ordinateur pendant que vous n'y êtes pas !

Pour terminer votre session, cliquez sur le menu  en haut à droite de l'écran, puis sur l'icône  dans ce menu et choisissez d'éteindre (*Power Off*) la machine (à choisir de préférence). Il est également possible de terminer la session sans éteindre la machine (pour qu'un autre utilisateur puisse l'utiliser juste après vous) : cliquez sur votre nom dans le menu  puis choisissez (*Log Out ...*, ou *Se Déconnecter*).

Exercice 4 (Logout) *Après avoir fermé les éventuelles fenêtres ouvertes, déconnectez-vous (pour ne pas perdre de temps, terminez simplement votre session sans éteindre/redémarrer la machine). Reconnectez-vous pour continuer les manipulations.*

2.2 L'environnement graphique

Quand vous vous identifiez sur une machine Unix (ou vous connectez à un serveur depuis un terminal), le premier programme qui démarre est un **environnement graphique** (aussi nommé **gestionnaire de fenêtres**, *window manager in english*).

Première remarque : il est sans doute différent de celui dont vous avez l'habitude. La majorité des ordinateurs grand-public fonctionnent avec le système d'exploitation Microsoft Windows, ou bien Mac OS X. L'ordinateur que vous utilisez maintenant fonctionne avec Unix, ou plus précisément GNU/Linux.



Le saviez-vous ?

Unix

Le mot **Unix** dans ce document est utilisé de manière générique pour désigner les systèmes d'exploitation proposant les mêmes interfaces de manipulations (héritées du système Unix d'origine).

Parmi les systèmes Unix les plus courants, citons GNU/Linux, Mac OS X, Solaris, FreeBSD, AIX.

Généralement, un environnement graphique sous Unix affiche des menus permettant de démarrer d'autres programmes. Il peut offrir des fonctions plus ou moins avancées, comme l'affichage de l'heure, de la météo, un calendrier, un contrôle des paramètres réseaux ou du son...

Il existe de très nombreux environnements graphiques pour les systèmes Unix, dont KDE, GNOME, XFCE, LXDE, IceWm, WindowMaker, Fvwm, TWM, Enlightenment...

2.2.1 L'environnement graphique : Gnome 3 Classic

L'environnement utilisé par défaut sur les PC est **Gnome 3 Classic**.



Le saviez-vous ?

Cela ressemble à Windows ?

En fait, ils ressemblent tous plus ou moins à leur ancêtre, le projet Alto, conçu au Xerox PARC, en 1973. Xerox n'a pas su l'exploiter commercialement. Steve Jobs, le fondateur d'Apple, après avoir visité le PARC en 1979, a lancé le *Macintosh* en 1979. Mais X11 permet aussi de s'éloigner du modèle classique de fenêtre à bandeaux et icônes, avec par exemple les familles de gestionnaires à *pavage* comme **ion3** ou **awesome**.

En haut de l'écran se trouve une barre de menus. Les menus permettant de lancer

des applications se trouvent à gauche (« *Applications* »). Ceux permettant de configurer l'ordinateur, le clavier et de fermer la session se trouvent sur la droite.

Exercice 5 (Lancer LibreOffice) *En utilisant la barre de menu de **Gnome**, lancez l'application **LibreOffice Writer**. Pour cela, cliquez sur **Applications** pour dérouler le menu.*

En mettant la souris dans le coin supérieur gauche, vous basculez vers la partie *Activités* qui vous permet également de passer d'une fenêtre à l'autre rapidement, de garder vos applications préférées dans les boutons de gauches et de les grouper en plusieurs *Bureaux Virtuels* à droite.

2.2.2 Quelques exercices avec le clavier Azerty

L'application **LibreOffice Writer** que nous venons de lancer est un traitement de texte. Nous allons en profiter pour voir quelques subtilités du clavier **Azerty**, utiles surtout pour les étudiants étrangers qui ne l'auraient jamais utilisé.

Le clavier Azerty permet de saisir des caractères accentués. La plupart sont disponibles directement sur le clavier (par exemple, « é » accessible sur la touche ) , mais il faut parfois utiliser des touches mortes, comme  : pour saisir un « ê », appuyez successivement sur  et sur  . Pour saisir le caractère « ^ » lui-même, appuyez sur  puis sur .

Les caractères les moins fréquents en Français (mais malheureusement assez présents en informatique) sont accessibles via la touche  , à droite de la touche  , et sont en général dessinés en bas à droite des touches (par exemple, « @ » est accessible via  +  et le caractère « ^ » s'obtient également via  + ).

Exercice 6 (Le clavier (exercice difficile pour œ)) *Essayez d'écrire « àêèïôûâæÉÊÛ » dans la fenêtre de **LibreOffice**. Pour vous y aider, vous devez utiliser les touches  et  , représentée par une flèche dirigée vers le haut, à droite et à gauche du clavier.*

2.3 Illustration de quelques différences entre Unix et Windows

Les points suivants montrent quelques fonctionnalités utiles, répandues et assez originales des gestionnaires de fenêtres Unix.

2.3.1 La souris a trois boutons !

Votre souris possède au moins trois boutons :

- celui de gauche, le bouton principal, utilisé pour la plupart des actions ;
- celui de droite, qui permet d'afficher un menu contextuel, c'est-à-dire que le menu est fonction de ce qu'il y a en dessous du pointeur de la souris ;
- celui du milieu (ou la molette), qui sert au copier-coller dans les applications. La molette de votre souris est un bouton que l'on peut appuyer normalement sans

faire tourner la molette. Pour les souris à deux boutons, on peut en général appuyer sur les deux boutons en même temps pour simuler le troisième bouton.

Exercice 7 (Le bandeau et les trois boutons !) *Essayez les différentes actions possibles en cliquant sur chacun des trois boutons sur le bandeau d'une fenêtre. Essayez aussi un clic droit sur le bandeau.*



Le saviez-vous ?

Faire tourner la molette d'une souris est identique à appuyer sur des boutons (les boutons 4 et 5). Il est donc possible de configurer des interactions du gestionnaire de fenêtre répondant à la molette.

2.3.2 Des astuces qui changent la vie

Le copier-coller juste avec la souris (bouton du milieu)

Ce point est sans doute l'un de ceux qui vous manqueront le plus dans les autres environnements.

La plupart des applications Unix graphiques acceptent un copier-coller « à la Windows » :

1. sélectionner à la souris ce que l'on veut copier ;
2. faire  +  , ou faire *Copier* dans le menu de l'application ;
3. mettre la souris à l'endroit où l'on veut coller ;
4. cliquer avec le bouton gauche de la souris pour mettre le curseur de l'application au bon endroit ;
5. faire  +  , ou faire *Coller* dans le menu de l'application.

Sous Unix, il y a plus simple et plus rapide !

1. sélectionner à la souris ce que l'on veut copier ;
2. mettre la souris à l'endroit où l'on veut coller ;
3. cliquer avec le bouton du milieu de la souris.

Exercice 8 (Le copier-coller à la souris) *Lancez un navigateur (firefox ou chrome)(en utilisant le menu « Activités » de Gnome), et une fenêtre LibreOffice si vous n'en avez pas déjà une ouverte. Essayez de copier-coller depuis votre navigateur vers LibreOffice, en utilisant les deux méthodes.*

Gestion des fenêtres, utilisation des touches *Alt*, *Ctrl* et *Majuscule*

Certaines touches ou combinaisons de touches permettent de modifier le comportement des interactions de la souris avec les applications ou le gestionnaire de fenêtres.

Exercice 9 (Déplacer et redimensionner les fenêtres avec Gnome) —

- Vous pouvez déplacer une fenêtre en maintenant enfoncée la touche  (à gauche de la barre espace), et en cliquant avec le bouton gauche **n'importe où** sur la fenêtre (pas uniquement sur le bandeau).*
- *Vous pouvez redimensionner une fenêtre en appuyant sur  et le bouton du milieu (ou le bouton droit, suivant les gestionnaires), **sans avoir besoin d'attraper un coin !***
 - *On peut maximiser une fenêtre en double-cliquant sur son bandeau, ou bien en faisant glisser ce bandeau vers le haut de l'écran.*
 - *On peut obtenir une fenêtre qui couvre la moitié gauche (resp. droite) de l'écran en attrapant son bandeau à la souris et en le faisant glisser contre le bord gauche (resp. droit) de l'écran.*
 - *On peut passer en arrière-plan une fenêtre avec un clic du milieu sur sa barre de titre.*



Le saviez-vous ?

Les touches ,  et  sont appelées, en terme X11, des modificateurs. , les deux , les deux  (quatre touches différentes, à droite et à gauche du clavier), ,  (la touche Super de son vrai nom) et  sont aussi des modificateurs qui peuvent être utilisés pour configurer des interactions.

2.3.3 Les bureaux virtuels

Les bureaux virtuels permettent de grouper les fenêtres et de pouvoir passer facilement d'un groupe à l'autre. Par exemple, il est possible d'utiliser un bureau pour lire son courriel et un autre bureau pour éditer le code d'un programme.

Exercice 10 (Les bureaux virtuels (Workspace)) *Il est possible d'ouvrir les fenêtres dans différents bureaux virtuels, accessible sous **Gnome** à droite dans les Activités (tout à droite dans l'écran gérant les activités), ou avec les raccourcis claviers  +  +  ou . Le menu contextuel obtenu par un clic droit sur le bandeau permet de déplacer les fenêtres d'un bureau à l'autre*
Essayez d'ouvrir plusieurs fenêtres et mettez-les dans des bureaux différents.



Le saviez-vous ?

L'environnement Gnome 3 ajoute automatiquement un bureau virtuel supplémentaire aux bureaux virtuels que vous utilisez.

2.3.4 Le terminal et le shell

Un des principes d'Unix consiste à avoir un outil (un programme) pour chaque tâche, et qui fasse cette tâche très bien.

Ces outils portent le nom de **commandes**.

Pour pouvoir utiliser ces commandes (les **exécuter**), il faut utiliser un programme particulier appelé **interpréteur de commande**.

C'est une interface textuelle qui lit les commandes tapées au clavier, les interprète et lance leur exécution. Concrètement, c'est une petite fenêtre toute simple qui affiche un *prompt* (début de ligne non modifiable en face duquel vous tapez vos commandes) ainsi que ce que vous tapez. L'interpréteur est également appelé *shell*.



Le saviez-vous ?

Techniquement, ce qu'on appelle le **terminal**, ou **xterm**, est la fenêtre qui gère l'affichage et la saisie de texte (*prompt*, commandes, résultats des commandes exécutées...). L'interpréteur de commande, ou *shell* est l'application qui s'exécute à l'intérieur et exécute vos ordres. Le même *shell* pourrait tourner sur une machine entièrement en mode texte. En fait le terminal est un émulateur de machine en mode texte, d'où son nom complet *X Terminal Emulator*.

Les habitués du système Windows confondent souvent le *shell* Unix et MS-DOS. Il n'en est rien, et le seul point commun entre les deux est le fait d'avoir une interface textuelle. Vous verrez que la ligne de commande sous Unix est en fait un outil très puissant, et la plupart des utilisateurs d'Unix l'utilisent au quotidien. Au point que les versions récentes de Windows proposent un *PowerShell* qui se rapproche du shell Unix.

Les commandes de base seront discutées au chapitre 7. Le *shell* exécute votre commande lorsque vous tapez sur .

Exercice 11 (Ouvrir une fenêtre avec un *shell*) Dans *Gnome*, on peut lancer un terminal depuis l'écran Activités. Le terminal se trouve dans les icônes de gauche Favoris. L'icône permettant de lancer un shell ressemble à un écran d'ordinateur avec une invite de commande (>).

On peut ajouter une commande aux Favoris en cliquant avec le bouton droit sur l'icône.

On peut aussi lancer une commande par son nom. En tapant  + , on obtient une petite invite de commande dans lequel on peut taper **gnome-terminal**. Lancez un shell et tapez dans le terminal la commande **cal** puis .

Vous pouvez maintenant fermer le shell. Notre shell étant dans une fenêtre graphique, il est possible de simplement fermer la fenêtre via la croix en haut à gauche.

Une méthode plus générale est de taper dans le terminal la commande **exit** puis , ou bien encore plus rapide : taper  + .

**Le saviez-vous ?**

Une autre solution pour obtenir un *shell*, consiste à ouvrir une nouvelle session en mode texte. Sous Linux, on peut faire cela avec `Ctrl` + `Alt` + `F2` (ou `F3`, `F4` ...), pour ouvrir des sessions textes sur plusieurs terminaux virtuels en même temps (sur certains PC, ça ne marche pas parce que le driver de la carte graphique ne sait pas passer en mode texte, on obtient alors un écran noir). On peut ensuite revenir à la session graphique avec `Ctrl` + `Alt` + `F1` (ou `F7`, selon les systèmes). Attention, on peut circuler d'un terminal virtuel à l'autre, mais les sessions restent ouvertes. Si vous avez lancé une session texte sur le terminal virtuel associé à `Ctrl` + `Alt` + `F2`, et que vous êtes revenus en mode graphique, votre session texte est encore ouverte, et restera accessible même après avoir terminé la session graphique (donc permettra potentiellement à un utilisateur mal intentionné d'utiliser votre compte une fois que vous serez parti de la machine). Pensez donc à terminer toutes vos sessions, avec la commande `exit` ou bien la séquence de touches `Ctrl` + `d`.

Tous les programmes que vous avez lancés depuis l'interface graphique sont également accessibles depuis ce terminal texte; chaque bouton de menu d'un environnement de bureau n'est en réalité rien de plus qu'un **lanceur** d'une commande prédéfinie (l'icône sur laquelle vous venez de cliquer exécute la commande `gnome-terminal` ou `xterm`).

Exercice 12 (Commande) Dans un terminal, tapez la commande `firefox` : elle ouvre un navigateur web. Essayez maintenant `oowriter` (ou `lowriter` si ça ne marche pas), vous devriez reconnaître *LibreOffice* que l'on a utilisé un peu plus haut.

**Le saviez-vous ?****Les processus gérant l'environnement**

La plupart des environnements **riches** (KDE, GNOME, XFCE...) sont gérés par plusieurs processus indépendants mais coopérants. Il est donc possible d'utiliser en même temps le gestionnaire de fichiers et d'icônes de GNOME (`nautilus`), la barre des tâches de XFCE (`xfce4-panel`) et le gestionnaire de fenêtre de KDE (`kwin`).

2.4 Règles élémentaires de sécurité

Un ordinateur est un outil de travail puissant, mais peut aussi permettre une utilisation malveillante, voire illégale ou criminelle dans des cas extrêmes. Un bon usage et une protection efficace de votre identité peuvent donc vous éviter de sérieux ennuis, car vous êtes pénalement responsables de votre utilisation des moyens informatiques.

2.4.1 Les risques

Votre compte informatique contient des données personnelles et vous offre un accès à Internet et à certaines ressources de calcul. Il sert aussi à vous identifier. Les attaques qu'une personne mal intentionnée peut mettre en œuvre à votre encontre peuvent prendre par exemple les formes suivantes :

- usurpation d'identité pour se livrer à des actes malveillants (transactions douteuses, piratage, *e-mails* injurieux, téléchargements illégaux...) dont vous seriez tenu responsable ;
- diffusion de virus (par *e-mail* notamment) ;
- fouille de vos fichiers ou mails personnels ;
- vol (copie) de votre travail (comptes-rendus, rapports, brouillons...) ;
- utilisation de vos ressources informatiques à votre insu ;
- ...

Bien entendu, cette liste n'est pas exhaustive et vise seulement à vous sensibiliser à l'importance de la sécurité informatique. Ces attaques sont monnaie courante, et si les techniques de sécurisation informatiques sont de plus en plus perfectionnées, elles peuvent être ruinées par l'imprudence d'utilisateurs se faisant pirater leur compte.

Par ailleurs, mal protéger votre compte, c'est aussi potentiellement permettre à des individus extérieurs de s'introduire dans le réseau local de l'école, et potentiellement d'attaquer d'autres machines. La sécurité de l'ensemble des systèmes de l'Ensimag repose donc sur l'application des bonnes pratiques par chacun d'entre vous.

2.4.2 Un vrai mot de passe

Le mot de passe qui vous a été attribué n'est certainement pas sûr, puisque vous n'avez même pas la garantie d'être le ou la seul(e) à le connaître. Il faut donc en changer.

Mais avant de choisir un nouveau mot de passe, il faut savoir ce qu'est un bon mot de passe. Ici, bon signifie robuste, comme dans « un bon cadenas ». Comme pour les cadenas, il y a toujours un moyen d'ouvrir sans avoir de clef. Si vous détenez des secrets d'État, ne les stockez pas sur votre compte Unix. Cela n'est pas seulement dû aux mots de passe ; beaucoup d'aspects du fonctionnement d'Unix le rendent « fragile » aux yeux de certains. Mais rassurez vous, il reste néanmoins un système beaucoup plus sûr que bien d'autres.

Un mot de passe sûr est un mot de passe difficile à deviner !

En effet, les logiciels de « cassage » de mot de passe procèdent par devinettes, en s'aidant de dictionnaires et d'ensemble de règles. Donc les noms propres et les noms communs sont de très mauvais mots de passe, quelle que soit leur origine. Le pire choix est d'utiliser son login comme mot de passe : c'est souvent le premier mot testé par les logiciels de cassage.

Parmi les erreurs à ne pas faire : choisir un mot de passe trop court. Un mot de passe de moins de 8 caractères peut être considéré d'office comme fragile. D'un autre côté, sur les machines de l'Ensimag, les mots de passes sont limités à 24 caractères.

Un bon mot de passe est donc un mot de passe d'au moins 8 caractères. alpha-numérique (lettre et chiffre).

Pour des raisons de limitations de certains services web universitaires, vous ne pouvez pas utiliser des caractères accentués, ni les caractères tels que @#&%_() []{}+*/. Uni-

quement des lettres (majuscules et minuscules) et des chiffres. Il faut au moins 1 chiffre, 1 majuscule et 1 minuscule.

Enfin, il faut que vous puissiez le mémoriser facilement.

Une stratégie possible pour concilier ces contraintes apparemment contradictoires est la suivante.

- Prendre comme base une phrase facile à mémoriser, et retenir les initiales des mots de la phrase, après en avoir éliminé les mots de liaison (le, de, un...) s'il y en a trop. Imaginons un proustien qui choisit la phrase « Longtemps je me suis couché de bonne heure », ça lui donne `ljmscdbh`.
- Une autre alternative est de choisir quatre mots indépendants au hasard. Attention, les mots doivent bien être difficiles à deviner, et indépendants (connaître 3 mots ne doit pas aider à trouver le quatrième). Par exemple, « Ensimag Informatique Mathématiques Appliquées » serait un très mauvais choix. Un bon choix, si il n'était pas si connu, pourrait être « correct cheval pile agrafe ». Ce qui donne en remplaçant le i par un 1 et en mettant des majuscules aux mots : `CorrectChevalPileAgrafe`.
- Introduire ensuite quelques variations permettant soit de compliquer encore, soit d'amener le mot de passe à au moins 8 caractères.

Exemple de variations faciles à mémoriser.

- Remplacer *de* par 2, *a* par @, *et* par &... .
- Utiliser des majuscules. Dans l'exemple précédent, je peux choisir de mettre les initiales de l'auteur au début et de tronquer la fin, ça donne `MP1jmsc2`.
- Quand la base choisie est inférieure à 8 caractères, il est recommandé de la compléter avec des caractères qui ne soient pas des lettres. Exemple : vous avez pris comme base « Jules Vernes de la Terre a la Lune », ce qui donne `JV2t@1` après élimination des articles, ça fait 6 caractères, c'est trop court. On peut faire : `JV::2t@1` ou `JV->2t@1`...
- Il est déconseillé d'utiliser des caractères accentués (èàùëïÔÉÀÛÔ...) parce que vous ne pourrez pas toujours les taper facilement pour faire un accès à distance (cf. chap 10) depuis un pays non francophone.

Exercice 13 (Pré-requis) Vérifier que les deux touches `[Maj]` (cf. l'exercice de la section 2.2.2) et `[Verr num]` (pavé numérique) sont dans un état compatible avec les caractères qui vous intéressent.

Exercice 14 (Changer de mot de passe) Maintenant que vous avez choisi un mot de passe sûr, il faut l'indiquer à la machine. La procédure pour changer de mot de passe est disponible sur l'intranet de l'école, à l'adresse <https://intranet.ensimag.fr/passwords/>.



Le saviez-vous ?

La commande pour changer de mot de passe sous Unix est en principe `passwd`, mais la synchronisation de mots de passe entre les machines (PC, serveurs, machines virtuelles, intranet, ...) rend les choses plus complexes à l'ENSIMAG.

2.5 L'intranet de l'Ensimag

Vous connaissez sans doute déjà le site Web de l'Ensimag :

`http://ensimag.grenoble-inp.fr/`

Mais l'Ensimag dispose aussi d'un grand nombre d'outils accessibles uniquement depuis l'école, ou bien via votre mot de passe depuis n'importe où sur Internet. Cet intranet se trouve à l'adresse :

`https://intranet.ensimag.fr/`

Exercice 15 (Intranet) *Ouvrez un navigateur web (Firefox par exemple), et ouvrez la page de l'intranet de l'école.*

Parmi les outils que vous utiliserez régulièrement sur cet intranet figurent les suivants.

Ensiwiki : dont nous avons déjà parlé en section 1.4.

Chamilo : une plateforme de e-learning, une fois connecté vous aurez accès à une page recensant toutes les matières de l'Ensimag, où vos enseignants mettront à disposition des supports de cours.

Zenith : des trombinoscopes des étudiants de l'école.

ADE : l'application qui gère les emplois du temps.

TEIDE : une application de rendu des TP et projets.

Exercice 16 (Chamilo) *Dans votre navigateur, accéder à Chamilo via l'intranet. Connectez-vous et retrouvez ce cours en naviguant dans les cours de l'école. Le cours se trouve dans « Tronc Commun », « 1ère année », « Introduction à l'algorithmique et à la programmation ».*

Dans l'immédiat, nous allons nous intéresser au premier outil de la liste : le Wiki.

Exercice 17 (Navigateur) *Dans votre navigateur, ouvrez l'URL `https://ensiwiki.ensimag.fr/`. Depuis la page d'accueil, accédez à la section « Stage Unix de Rentrée » (un lien direct sera ajouté au cadre « Actualité » de la barre de navigation du site pendant la durée du stage).*

Vous trouverez une section consacrée au stage de rentrée, contenant des liens vers les pages du Wiki auxquelles ce document fait référence.

Si on parle d'EnsiWiki, « Premiers pas avec Emacs et Python », vous pouvez donc soit taper l'URL `https://ensiwiki.ensimag.fr/index.php/Premiers_pas_avec_Emacs_et_Python` (dans les URLs EnsiWiki, les « _ » et les espaces sont équivalents) directement dans votre navigateur, soit trouver le lien « Premiers pas avec Emacs et Python » dans la page du stage, et y accéder directement. Si vous êtes perdus, le champ « recherche » en haut de chaque page est votre ami.

Exercice 18 (Emacs et Python) *Ouvrez la section « Premiers pas avec Emacs et Python » du Wiki (nous la lirons un peu plus tard).*

2.6 Un problème ?

En cas de problème avec les machines de l'école, ou avec votre machine personnelle, vous pouvez contacter le service informatique. La procédure et les adresses emails à utiliser sont documentés sur EnsiWiki, « Dépannage à l'Ensimag ».

Chapitre 3

Internet et courrier électronique à l'Ensimag

3.1 Navigation sur le Web : Firefox

Vous le connaissez sans doute, et il gagne à être connu : **Firefox** est un navigateur web très pratique, rapide, et très bien sécurisé. S'il ne vous suffit pas, vous trouverez probablement votre bonheur parmi les milliers d'extensions disponibles ! Rendez-vous sur <https://www.mozilla.org/fr/firefox/> pour les détails.

Firefox est accessible via la barre de menus de **Gnome**, ou via la commande unix `firefox`.

D'autres bons navigateurs web sont disponibles sous Linux, comme **Chromium** ou **Opera**, mais ils ne sont pas forcément installés à l'Ensimag.

3.2 Votre adresse de courriel

Votre adresse a été choisie de manière canonique en utilisant le modèle suivant : `prenom.nom@grenoble-inp.org` . Les espaces présents dans les noms ou prénoms sont remplacés par des tirets (« - »). Les caractères accentués sont remplacés par le caractère non accentué le plus proche.



Le saviez-vous ?

Dans une adresse de courriel, les majuscules et minuscules ne sont, normalement, pas différenciés.

Pour résoudre les problèmes d'homonymies, de légères adaptations autour de ce modèle sont aussi réalisées au cas par cas.

Nous allons maintenant voir comment échanger des courriers électroniques avec n'importe quelle personne ayant également une « boîte aux lettres électroniques » sur Internet. À nouveau, nous avons besoin d'une application cliente pour recevoir, envoyer, et gérer notre courrier. Nous ne verrons pas ici apparaître les serveurs et les agents communicants, mais sachez qu'ils existent. Une fois encore, il y a de nombreux clients disponibles, certains graphiques, d'autres s'exécutant dans des terminaux textuels. Nous allons décrire ici trois clients : un client graphique, un client web, et un client texte.

Nous vous demandons d'essayer de vous envoyer à vous-même des courriels et de les lire avec les trois interfaces. Le but est ici de vous montrer les avantages et les faiblesses de ces trois approches.



Le saviez-vous ?

Adresses des enseignants

Les enseignants et intervenants à l'Ensimag ont, en grande majorité, une adresse électronique de la forme *pre-nom.nom@imag.fr*, *prenom.nom@univ-grenoble-alpes.fr* et/ou *prenom.nom@grenoble-inp.fr* (NB : **grenoble-inp.fr** et non **grenoble-inp.org** qui est utilisé pour les adresses à vie des étudiants).

Les emails, le téléphone ou l'adresse du bureau de vos enseignants sont dans l'annuaire des personnels des universités grenobloise. L'adresse de l'annuaire est indiquée dans Ensiwiki, « Portail:Ensimag#Liens_externes »

3.3 Netiquette

Qui dit droit dit également devoir. Vous avez signé une charte informatique donnant le règlement d'utilisation des ressources, mais le bon usage est plus pointilleux que ce document (de même que les bonnes manières comportent plus d'exigences que la loi).

Par exemple, vous n'apprécieriez sans doute pas qu'une personne transmette à toute la promo (sans votre avis) un e-mail personnel que vous venez de lui envoyer. Cela fait donc partie des choses à ne pas faire.

L'ensemble des règles de bon usage d'Internet forme ce qu'on appelle la **Netiquette** (contraction de « Net » et « étiquette »). Ces règles ont été normalisées au niveau international et il est de bon augure (et surtout prudent) de les respecter. Le document de référence est la RFC 1855 <https://tools.ietf.org/html/rfc1855>. Vous trouverez facilement sur Internet une traduction française de ce document. Nous ne mentionnons ici les règles les plus importantes concernant le courrier électronique.

3.3.1 Rédaction

- Remplissez toujours le champ **objet**, et ce le plus précisément possible. Votre destinataire reçoit peut-être un grand nombre de mails et doit pouvoir savoir immédiatement de quoi il est question.
- **Signez** vos courriels à la fin du texte, même si votre adresse e-mail est censée pouvoir être visible. Votre signature doit inclure votre nom complet et adresse e-mail.
- N'écrivez jamais **rien de confidentiel** dans un e-mail, et plus généralement rien que vous n'oseriez écrire dans une carte postale.
- Écrire tout en majuscules revient à CRIER. Pour insister sur un mot ou une phrase, entourez-le (la) du caractère * ou _, ***comme ceci***.
- N'écrivez pas en langage SMS, vous n'êtes pas limité à 160 caractères.
- Soyez bref (mais courtois), venez-en au fait.

-
- Manipulez l’humour et l’ironie avec prudence (ils passent mal par écrit et peuvent être mal interprétés, même avec un *smiley*).
 - Soyez toujours **modéré** et prudent dans vos propos, même si on vous provoque, car votre message peut être transféré (parfois partiellement, hors de tout contexte) et lu par n’importe qui.
 - Gardez toujours un **esprit critique** devant les e-mails que vous recevez : il est très facile de se faire passer pour n’importe qui.
 - Lorsque vous répondez à une personne, gardez si nécessaire la portion de mail à laquelle vous répondez, en effaçant ce qui n’est plus de circonstance ; cela évite d’avoir des mails très longs et illisibles au bout de quelques échanges.
 - Utilisez une adresse professionnelle pour les communications professionnelles (vous serez plus crédibles en écrivant à un enseignant depuis une adresse `@ensimag.grenoble-inp.fr` que depuis `toto_a_la_plage@hotmail.com`).
 - Limitez, si possible, à un sujet le contenu de vos messages.

3.3.2 Transfert, envoi à de multiples personnes

- Ne transmettez pas un courrier reçu à d’autres personnes sans l’autorisation de l’émetteur.
- Lorsque vous transmettez un message, n’en modifiez pas le contenu.
- Ne transmettez jamais de chaînes de mail.
- Lorsque vous recevez un mail destiné à une liste de diffusion, évitez de « répondre à tous » mais sélectionnez soigneusement vos destinataires. Faites toujours attention aux personnes en « copie carbone » des mails auxquels vous répondez.
- Lorsque vous envoyez un mail à beaucoup de personnes, utilisez de préférence le champ « copie cachée » (`cc`, ou `bcc`) pour les adresses e-mail, sauf si les personnes se connaissent et qu’elles doivent avoir connaissance des autres destinataires.
- Vérifiez toujours le(s) destinataire(s) avant d’envoyer un mail.
- Évitez de relayer des *hoax* ou canulars en français (alertes pour de faux virus, rumeurs, fausses informations). Afin de vérifier la véracité ou non d’une information véhiculée par un mail, vous pouvez vous rendre sur <http://www.hoaxbuster.com>. En pratique, un message qui vous demande de transmettre une information a tout votre carnet d’adresse a 99% de chances d’être un canular !

3.3.3 Pièces jointes

- **Limitez la taille** et le nombre de vos pièces jointes. En plus de surcharger les boîtes aux lettres de vos destinataires vous prenez le risque que votre message soit supprimé (sans notification) par un des serveurs de messagerie responsable de son acheminement si sa taille excède les limites autorisées. Utilisez la compression et archivage (section 7.3). De manière générale, évitez les messages de plus de 5 Mo. De plus sachez que les fichiers binaires subissent une expansion d’environ 33 % en raison d’un procédé de transcodage binaire → ASCII.
- Utilisez des formats **portables, libres d’utilisation**, et non spécifiques à une plateforme. À la place de `.doc` créés avec **Microsoft Office**, on préférera ainsi des documents au format PDF, qui préserve la mise en forme d’un document indifféremment du système d’exploitation ou de l’application utilisée pour le lire, ou encore le format ODF (`.odt` pour les textes) si le document doit pouvoir être

modifié par le destinataire.

- Quand c'est possible, privilégiez un lien vers un fichier.

3.4 Un client de messagerie simple à utiliser : Thunderbird

Thunderbird est un lecteur de courriel graphique classique mais très complet. Il est historiquement le compagnon de Firefox avec qui il a un certain nombre de points communs. En particulier il dispose lui aussi d'un grand nombre de modules d'extensions pour divers usages ou adaptations de l'interface.

Exercice 19 (Envoyez un courriel à votre voisin) *Demandez à un de vos voisins son adresse et envoyez-lui un courriel.*

3.5 Lire et envoyer des mails depuis n'importe où : le *webmail*

Lorsque vous êtes en déplacement, la méthode la plus simple pour lire et répondre à vos *mails* est d'utiliser n'importe quel navigateur Internet pour contacter le serveur *webmail* de l'Ensimag à l'URL : <https://webmail.grenoble-inp.org/> (l'ancien webmail, <https://etu-ensimag.mail.grenet.fr/>, est encore utilisé par les anciens comptes mais vous n'êtes probablement pas concerné si vous lisez ceci). Ce webmail utilise le logiciel Zimbra.



Attention !

Sécurité

Attention, le webmail est un site sur lequel vous allez entrer votre mot de passe, il est important de vous assurer que vous êtes sur le bon site! Ne **surtout** pas se contenter de chercher « mail ensimag » dans un moteur de recherche en cliquant aveuglément sur le premier lien, qui a de bonnes chances de ne pas être le bon : il faut connaître l'adresse (votre navigateur peut vous aider) ou passer par un lien depuis l'intranet.

Le nom d'utilisateur et le mot de passe sont ceux de votre compte INP.



Le saviez-vous ?

Limitations *webmail*

Un *webmail* ne vous permet pas de travailler hors ligne. Ainsi, des déconnexions intempestives peuvent conduire à la perte de votre projet de message. De plus, l'interface *webmail* est quand même relativement lourde et lente :

- l'interface n'est pas faite pour un usage intensif mais plutôt pour un accès d'appoint avec un trafic de courriel réduit.
- la quantité d'information transitant sur le réseau juste pour lire le texte de vos courriels est parfois non négligeable pour la bande passante dont vous disposerez.
- en général, un webmail ne vous donne pas d'accès à vos mails hors-ligne. Un client lourd permet de synchroniser les messages sur le disque local, et de pouvoir y accéder sans être connecté.
- Beaucoup d'utilisateurs apprécient de pouvoir choisir leur client lourd (Thunderbird, pine, ou leur client préféré sur Android ou iOS), alors que le webmail offre peu de choix de configuration.

Comme le *webmail* Zimbra de l'école, plusieurs clients lourds vous offrent des services collaboratifs (souvent standardisés) tel que la synchronisation des agendas, des carnets d'adresses... Si ces perspectives vous intéressent, vous pouvez vous renseigner sur les extensions **Lightning** et **SOG connector** pour Thunderbird, ou sur le projet **Evolution** de GNOME.

3.6 IMAP

Les deux lecteurs présentés (section 3.4, 3.5) utilisent le protocole IMAP pour gérer votre boîte aux lettres à distance. De manière plus générale, IMAP peut être considéré comme un protocole de synchronisation centralisé de boîtes aux lettres. Il en résulte que les messages que vous manipulerez seront les mêmes, quel que soit le client de messagerie utilisé.



Le saviez-vous ?

IMAP à l'Ensimag

Votre serveur de courriel IMAP est : webmail.grenoble-inp.org. Vous pouvez l'utiliser de n'importe où sur la planète. Dans ce cas il faut activer l'utilisation du chiffrement SSL pour éviter d'avoir votre mot de passe circulant en clair sur le réseau. Les détails de sa configuration se trouvent dans le Wiki, « Thunderbird ».



Le saviez-vous ?

Lire ses comptes courriels externes (Gmail, Free, Hotmail, Laposte.net...) à l'Ensimag

La plupart des services de courriels proposent un accès IMAP aux boîtes aux lettres. Vous pouvez donc ajoutez dans Thunderbird la configuration pour lire directement vos comptes (cf « Configurer thunderbird pour lire ses emails externes » d'EnsiWiki)).

Comme le protocole IMAP laisse les courriels sur le serveur distant, vous pouvez facilement alterner une lecture avec Thunderbird à l'Ensimag et l'interface de type *webmail* chez vous par exemple.

3.6.1 Sauvegarder ses courriels dans des dossiers

Il vous est possible de stocker votre courrier dans des dossiers. Pour que ces dossiers soient accessibles depuis n'importe où en IMAP, il est important de les créer exclusivement dans le dossier IMAP et jamais dans les répertoires locaux à la machine (*Local Folders*)

3.6.2 L'envoi est différent de la lecture

L'envoi de courriel passe par un protocole différent de la réception : SMTP, ou sa variante sécurisée SMTPS.

L'accès SMTPS est visiblement bloqué sur `webmail.grenoble-inp.org`, donc nous n'avons pas pour l'instant d'accès SMTP(S). Vous pouvez au choix utiliser le webmail pour envoyer vos messages, ou bien utiliser celui de votre fournisseur d'accès (ou d'email) habituel.



Le saviez-vous ?

Envoi avec un client texte

En fait, si vous êtes connectés de l'extérieur sur un serveur de l'école (par exemple via SSH), vous pouvez parfaitement utiliser les clients textuels présentés dans la section 3.7 pour envoyer vos courriels, puisque vous travaillez sur une machine de l'Ensimag.

3.7 Les lecteurs de courriel textuels : Alpine, Mutt

Les courriels contiennent principalement du texte. Il existe donc de nombreux clients textuels dont l'avantage principal est la rapidité et la facilité d'utilisation à travers une connexion SSH.

La différence est notable par rapport au *webmail* lorsque votre volume de courriel devient important. Le fait qu'ils ne soient pas graphiques ne signifie pas qu'ils sont plus difficiles à utiliser. Ils affichent directement dans le terminal la liste des touches de commandes qui ont une action pertinente à un instant donné.

Alpine est un lecteur de courriel convivial et qui vous guide beaucoup dans son utilisation. Il demande néanmoins un peu de configuration si vous êtes en dehors de l'Ensimag. **Mutt** est un peu plus minimaliste mais son usage depuis n'importe où pour lire votre courriel en IMAP sécurisé est aussi simple que la commande :

```
mutt -f imaps://prenom.nom@grenoble-inp.org@webmail.grenoble-inp.org
```

(Votre login est votre email, `prenom.nom@grenoble-inp.org`, d'où la présence de deux @ dans la ligne de commande)



Si vous n'avez pas encore commencé le jeu de piste, rendez-vous sur EnsiWiki, « TP Unix - Jeu de piste ». Si vous avez bien lu le document jusqu'ici, vous devriez pouvoir terminer les étapes A1 à A5, et atteindre l'énoncé de l'étape B1. Si vous bloquez sur une étape, discutez-en avec d'autres étudiants et/ou demandez de l'aide à vos enseignants.

Chapitre 4

Le b.a.-ba pour survivre sous Unix

4.1 Système d'exploitation

Un ordinateur est composé d'un ensemble d'unités matérielles, chacune destinée à réaliser une tâche particulière. Que ce soit le processeur et la mémoire qui servent à exécuter les logiciels et à « piloter » l'ordinateur complet (comprenez par là : demander des services aux autres composants), les éléments d'interface avec un humain (clavier, souris, écran...), de persistance de données (disque dur...), ou encore l'infrastructure de communication entre ces composants et avec d'autres machines (interface réseau, USB...), tous ces composants nécessitent d'être gérés et éventuellement partagés entre plusieurs utilisateurs : c'est le rôle du **système d'exploitation**. Ce dernier est généralement constitué d'un **noyau** (Linux par exemple) et d'un ensemble de logiciels permettant aux utilisateurs d'assurer la gestion de l'ordinateur et des programmes et applications exécutés.

Certains constructeurs d'ordinateur possèdent leur système d'exploitation « maison », mais dans le domaine de la micro-informatique, des systèmes standard sont majoritairement utilisés. Windows et la famille Unix (MacOS, GNU/Linux [fn : : Le « GNU » représente dans cette dénomination l'ensemble des commandes Unix du projet GNU s'exécutant sur le noyau « Linux ». Voir section 5.7 pour plus de détails sur ce projet.]...) sont les plus répandus.

4.2 Commandes et interpréteur de commandes

4.2.1 Notion de commande

Nous allons parler ici de la fraction du système d'exploitation que nous avons définie comme permettant la gestion de l'ordinateur et des applications. Dans Unix, ces services sont réalisés par des programmes portant le nom de **commande**, que nous avons déjà vues en section 2.3.4. Certaines commandes, particulièrement simples, ne nécessitent pour être exécutées, que la frappe de leur nom sur le clavier, dans un terminal.

Lancez un **shell** (c'est-à-dire un *terminal*). Tout ce qui suit dans ce chapitre se fera dans cette fenêtre.

Exemple :

Unix comporte une commande permettant d'afficher un calendrier. Par défaut, le mois courant est affiché. Cette commande porte le nom de `cal`. Lancez-la en tapant sur votre

clavier :

`cal`

puis `[entrée]` .

Certaines commandes nécessitent ou permettent qu'on leur fournisse des paramètres lors de leur lancement. L'exécution d'une telle commande se fait en tapant sur la même ligne le nom de la commande suivi de ses paramètres, comme ceci :

```
nom-de-la-commande param_1 param_2 ... param_n
```

puis `[entrée]`

On parle aussi souvent d'« argument » à la place de paramètre. Ce document utilise « argument » et « paramètre » indifféremment.

La même commande, `cal`, permet aussi d'imprimer le calendrier d'un mois quelconque d'une année comprise entre 0 et 9999. Au lancement, il faut lui fournir en paramètre le numéro du mois et de l'année choisis. Pour vous familiariser avec son emploi, tapez par exemple :

```
cal 2 1998
```

puis `[entrée]`

puis affichez le mois de votre naissance.

A partir de maintenant, nous omettons `[entrée]` dans les exemples de commandes : vous savez qu'il faut valider une commande en appuyant sur cette touche pour qu'elle s'exécute.

N'hésitez pas à consulter l'aide de `cal` avec `man cal`.

4.2.2 Notion d'interpréteur de commande

Nous avons découvert l'interpréteur de commande de un peu plus haut, en section 2.3.4. Il est maintenant temps d'apprendre à s'en servir réellement. Contrairement aux interfaces graphiques, cet apprentissage va demander un certain investissement : au minimum, vous aurez à connaître quelques dizaines de commandes utiles dans la vie de tous les jours, et pour les ambitieux qui voudraient les connaître toutes, sachez qu'il en existe au total plusieurs milliers.

Au fil de votre apprentissage, vous découvrirez que dans bien des cas, un interpréteur de commande est plus efficace qu'une interface graphique, voire permet d'accéder à des fonctionnalités qui ne sont pas accessibles à la souris.

L'interpréteur de commandes réalise le travail suivant :

1. écriture de l'invite de commande ou *prompt*. Il s'agit d'une courte chaîne de caractères généralement terminée par le caractère `%` ou `$`, et qui sert à inviter l'utilisateur à taper une commande ;
2. lecture sur le clavier d'une ligne tapée par l'utilisateur et terminée par `[entrée]` ;
3. interprétation du premier mot comme étant le nom d'une commande, et des autres mots comme étant les paramètres de la commande ;
4. exécution de la commande désirée ;

5. retour à l'étape numéro 1.

Exercice 20 (Commande incorrecte) *Que se passe-t-il si l'utilisateur tape un nom de commande incorrect ? Tapez par exemple :*

xyz32

et mémorisez bien le message d'erreur : c'est celui que vous donnera l'interpréteur de commandes à chaque fois que vous taperez quelque chose qui ne correspond à aucun des noms de commande qu'il connaît.

L'intérêt des commandes et de l'interpréteur de commandes est à la fois sa rapidité et sa puissance. Les commandes à connaître absolument sont décrites au chapitre 7. En attendant, un petit exemple d'utilisation peut vous donner une idée de l'efficacité du shell.

Exemple :

On souhaite rechercher un fichier contenant le mot « Factorielle » mais on ne connaît plus son nom. La ligne de commande suivante permet d'identifier tous les fichiers contenant la chaîne de caractères « Factorielle » :

```
grep "Factorielle" *
```

La réponse donnée par l'exécution de cette commande affiche pour chaque fichier contenant cette chaîne le nom du fichier suivi de la ligne où la chaîne « Factorielle » a été trouvée. Par exemple :

```
fact.py:      def fact(n):      # Factorielle
```

4.3 Le système de fichiers d'Unix

Nous allons maintenant partir à la découverte de l'arborescence de répertoires d'Unix.

4.3.1 Notions de fichier et de répertoire

Les fichiers peuvent être manipulés à l'aide de commandes dans un shell (cf. section 4.2.2) ou bien à l'aide d'une application particulière : un gestionnaire de fichiers graphique. Dans ce chapitre, nous utiliserons simultanément le shell et le gestionnaire graphique *nautilus*¹, afin de bien appréhender l'utilisation du système. Il est important de se rendre compte que ces deux outils fournissent deux vues différentes **de la même information**.

Le côté « graphique » de *nautilus* est attrayant, et sa simplicité d'utilisation peut séduire, mais vous apprendrez progressivement à vous servir de la ligne de commande, qui, avec de l'expérience, est en fait **beaucoup** plus efficace pour la plupart des tâches. Dans tous les cas, il sera indispensable d'apprendre à utiliser la ligne de commande. Après avoir fait connaissance avec le système, nous vous demanderons donc d'arrêter d'utiliser *nautilus* pendant au moins quelques mois.

1. <https://live.gnome.org/Nautilus>

4.3.2 Un gestionnaire de fichiers graphique : nautilus

Sous Unix, il existe une grande variété de gestionnaires de fichiers graphiques, chacun ayant ses avantages et ses inconvénients. Dans ce chapitre nous utiliserons **nautilus**.

D'autres choix étaient presque équivalents comme **Thunar** ou **Konqueror**.

En parallèle, nous utiliserons un shell. Le but est d'illustrer un certain nombre de concepts avancés des fichiers sous Unix. Il faudra déplacer et redimensionner les fenêtres l'une à côté de l'autre afin de voir et manipuler simultanément **nautilus** et le terminal.

Exercice 21 (Démarrez nautilus) *Pour lancer nautilus, cliquez sur l'icône « Home » (ou « Computer ») sur le bureau, ou choisissez un répertoire dans le menu « Raccourcis » (ou « Places »).*

Par défaut, nautilus affiche le répertoire courant de manière conviviale, mais nous avons besoin de voir et de comprendre ce qu'il se passe. Faites  +  sur une fenêtre nautilus : le répertoire est maintenant affiché sous forme texte (par exemple, /user/1/toto), et réciproquement, vous pouvez entrer le nom d'un répertoire sous forme texte.

Vous pouvez activer ce mode définitivement en entrant la commande suivante dans un terminal :

```
gsettings set org.gnome.nautilus.preferences always-use-location-entry true
```

Exercice 22 (Équivalent shell) *Démarrez un terminal. Le prompt correspond au nom de votre répertoire personnel. Pour afficher le contenu de ce répertoire, tapez `ls`. Cette commande (en anglais list) vous permet d'afficher les fichiers visibles du répertoire où vous vous placez.*

Vous devriez voir les mêmes fichiers depuis nautilus et dans le terminal.



Le saviez-vous ?

nautilus

Il propose des possibilités d'accès à distance, entre autres, avec **ftp** (transfert de fichiers), **ssh** (protocole réseau crypté), **smb** (partage de fichiers Windows). Essayez d'ouvrir l'emplacement `sftp://pcserveur.ensimag.fr/` par exemple.

4.3.3 Définitions

Pour le stockage de l'information, l'informatique s'est inspirée des systèmes de stockage « papier » : un fichier informatique peut être comparé à une feuille de papier, et on peut regrouper ces feuilles en **dossiers** (ou *folders* en Anglais, qui tient son nom du fait qu'on fabrique un dossier papier en pliant une feuille de carton), ou **répertoire** (ou *directory* en anglais). Bien évidemment, un répertoire peut contenir non seulement des fichiers, mais aussi d'autres répertoires.

- Un **fichier** est l'unité de stockage de l'information dans l'ordinateur. Il est identifié par un nom et un chemin d'accès.
- Un **répertoire** est une unité d'organisation des fichiers. Chaque répertoire peut contenir des fichiers et/ou des répertoires. Il est identifié par un nom et un chemin

d'accès.

- Le **chemin d'accès** à un fichier (ou à un répertoire) est la suite des répertoires imbriqués qu'il faut ouvrir pour accéder à ce fichier (ou répertoire).

Chaque utilisateur possède un répertoire personnel (« home directory » en anglais), où sont stockées toutes ses données. Traditionnellement sous unix, le répertoire personnel de l'utilisateur `toto` se trouve dans `/home/toto/`, mais l'Ensimag utilise une disposition de répertoires un peu plus compliquée. Mais il n'est pas nécessaire de savoir où se trouve le répertoire : on peut désigner le répertoire personnel de l'utilisateur `toto` avec le raccourci `~toto`, et dans le cas particulier où un utilisateur souhaite accéder à son propre répertoire personnel, on peut utiliser le raccourci `~`.



Attention !

Subtilités avec la notation `~`.

Une confusion classique est d'écrire `~user` à la place de `~/user`, ou l'inverse. `~user` signifie « le répertoire personnel de `user` », alors que le `/` dans la notation `~/user` est le séparateur de répertoire classique, donc `~/user` signifie « le répertoire nommé `user` à l'intérieur du répertoire `~` (qui est le répertoire personnel de l'utilisateur courant) ».

Exemple :

Le chemin `/user/1/perronni/` est le chemin d'accès au répertoire personnel de l'utilisateur `perronni`. Pour accéder à ce répertoire il faut donc « ouvrir » le répertoire `/` (le répertoire racine) qui contient le répertoire `users` qui contient le répertoire `1` qui à son tour contient le répertoire `perronni`.



Attention !

Majuscules/minuscules.

Pour les noms de fichiers ou de répertoires les minuscules et majuscules ne sont pas équivalentes.

Sous Unix, chaque utilisateur a un répertoire (ou espace) personnel, et n'a en général pas accès aux répertoires des autres utilisateurs (ou au moins, n'a pas l'autorisation d'y écrire).

Exercice 23 (nautilus et les répertoires) *Dans la fenêtre nautilus, une barre (dite de localisation) en haut à droite devrait afficher quelque chose comme `/user/1/votre-login/`. Il faut lire cela comme : « dans l'ensemble des fichiers, dans le répertoire intitulé `users`, dans un sous-répertoire intitulé `votre-login`... » Essayez par exemple d'effacer la fin de cette ligne pour afficher seulement `/user/1/` (remplacez `1` par le numéro correspondant qui s'est affiché quand vous avez affiché le nom complet de votre répertoire personnel), puis validez avec . Vous voilà maintenant dans un répertoire parent de votre répertoire personnel, qui contient votre répertoire et ceux des autres étudiants.*

Exercice 24 (Changer de répertoire dans le terminal) *Pour effectuer la même manœuvre dans le terminal, vous pouvez utiliser la commande `cd` qui signifie « Change Directory » : tapez `cd /user/1/`. Vous êtes maintenant dans le répertoire parent de votre répertoire personnel. Vous pouvez afficher son contenu en tapant `ls` et constater que vous observez la même chose que dans `nautilus`. Pour revenir à votre espace personnel, vous pouvez taper `cd ~votre-login`. Si vous êtes perdus, la commande `cd`, sans argument, permet de revenir dans votre répertoire personnel.*



Le saviez-vous ?

Une seule arborescence

Un système Unix utilise une seule arborescence, quel que soit le nombre de périphériques de stockage présents ou de disques réseaux accédés. Toutes les données sont **montées** quelque part dans cette arborescence. Par exemple, un DVD de données contenant un répertoire `toto` peut être monté dans le répertoire `media` en tant que `/media/dvd/`. Le répertoire `toto` est alors accessible en `/media/dvd/toto`. La liste courante des points de montage est consultable avec la commande `mount`.

Chemin absolu et relatif

Dans l'exercice précédent, une fois que vous êtes dans le répertoire parent de votre répertoire personnel, vous pouvez « voir » votre répertoire personnel en tapant `ls` (ou visuellement avec `nautilus`). Lorsque vous tapez la commande `cd`, vous lui donnez en argument le **chemin** où vous voulez vous placer. Ce chemin peut aussi être **relatif**, auquel cas vous devez seulement préciser la liste des répertoires à parcourir en partant du lieu où vous vous trouvez. C'est ce que vous avez fait lorsque vous étiez dans `/user/1/` et que vous avez tapé `cd ~votre-login`.

Par ailleurs, le chemin d'accès est dit **absolu** lorsqu'il contient la totalité des répertoires d'inclusion, par exemple lorsque vous avez tapé `cd /user/1/`. L'avantage de cette notation est qu'elle est plus rapide si vous voulez accéder à un répertoire éloigné de celui où vous vous trouvez.

Exercice 25 (Changer de répertoire dans le terminal (chemin relatif))

Pour effectuer la même manœuvre dans le terminal, avec un chemin relatif, tapez `cd ..` (avec un espace et deux points successifs : le répertoire parent). Vous êtes maintenant dans le répertoire parent de votre répertoire personnel. Vous pouvez afficher son contenu en tapant `ls` et constater que vous observez la même chose que dans `nautilus`. Pour revenir à votre espace personnel, vous pouvez aussi taper `cd ../votre-login/` (1 seul point : le répertoire courant).

Exercice 26 (Changer de répertoire dans `nautilus` (chemin relatif))

Essayez d'entrer et de comprendre le chemin suivant dans la barre de localisation : `/user/1/votre-login/../../votre-login/../../votre-login/../../votre-login`

4.3.4 Naviguer dans l'arborescence

Dans *nautilus*, il suffit de cliquer (double-clic, ou 1 clic simple selon la configuration) sur un répertoire pour le visiter. Pour remonter d'un niveau, on peut éditer le chemin sur la barre de localisation s'il est en mode texte, ou cliquer sur le répertoire parent s'il est en mode graphique. Certaines versions de *nautilus* ont un bouton ↑ pour monter d'un niveau.

Exercice 27 (Changer de répertoire) *Essayez de vous déplacer dans un sous-répertoire de votre répertoire courant puis de revenir.*

4.3.5 Déplacer, copier, supprimer des fichiers

Pour manipuler des fichiers sous *nautilus*, on commence par les sélectionner. On peut cliquer sur un fichier pour le sélectionner. Pour sélectionner plusieurs fichiers à la fois, on peut dessiner un rectangle de sélection en faisant glisser la souris, ou bien utiliser les touches  et .

Exercice 28 (Sélection de fichiers) *Cliquez sur le fichier en haut à gauche de la fenêtre *nautilus*, puis sur un fichier vers la droite de la fenêtre. Recommencez, mais en maintenant la touche  appuyée pendant que vous sélectionnez le deuxième fichier. Même chose avec .*

Une fois les fichiers sélectionnés, on peut les déplacer ou les copier en faisant glisser la souris vers la destination. Pour les effacer, un clic droit ouvre un menu contextuel proposant l'effacement.



Attention !

L'effacement dans un gestionnaire de fichier graphique n'est souvent pas immédiat. Comme sous d'autres systèmes, les fichiers sont déplacés dans la corbeille. Il faut vider la corbeille pour réellement effacer le fichier.



Le saviez-vous ?

Votre corbeille est un répertoire de votre compte comme les autres : `~/.local/share/Trash/files`. Son nom a été fixé par convention entre les différents gestionnaires de fichiers graphiques.

4.3.6 L'arborescence de répertoires usuelle sous Unix

Notion de répertoire d'origine

Nous avons vu que chaque utilisateur possède dans le système un ensemble de fichiers et d'arborescences, et bien évidemment le système est obligé de gérer tous ces objets de manière séparée pour chaque utilisateur. Pour cela, tous les objets appartenant à un utilisateur sont reliés à un même répertoire. Ce répertoire est nommé le **répertoire**

d'origine de l'utilisateur (en anglais, *home directory*). Lorsque vous vous connectez (*login*), le système fait initialement correspondre votre répertoire courant à votre répertoire d'origine. Vous ne voyez ainsi que les objets qui vous appartiennent. Le répertoire d'origine est souvent représenté graphiquement par une petite maison.

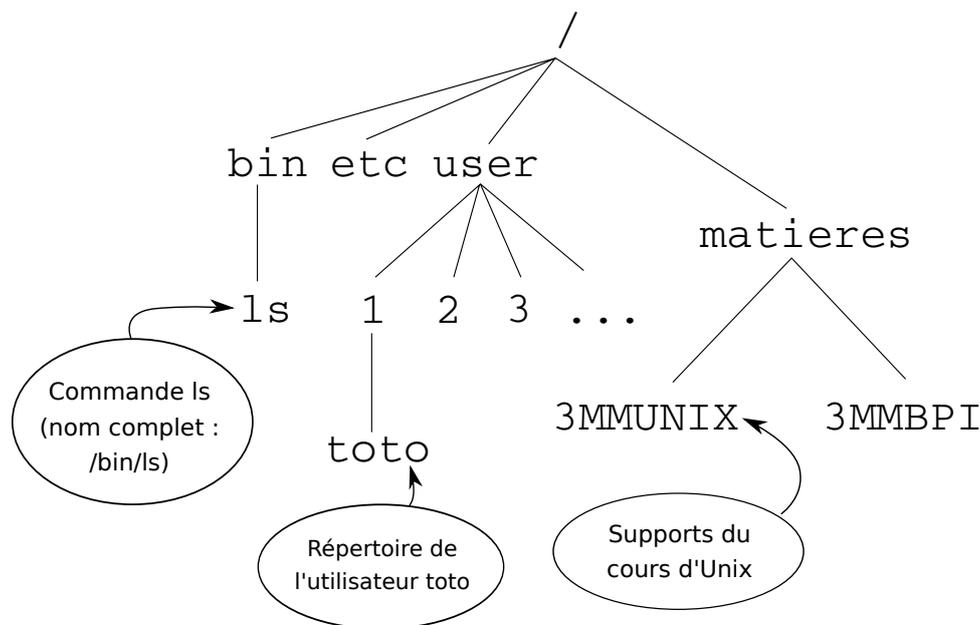
Exercice 29 (Retournez au répertoire d'origine dans le terminal)

Essayez de vous déplacer dans le répertoire /tmp. Retournez ensuite dans votre répertoire d'origine.

Exercice 30 (Retournez au répertoire d'origine dans nautilus) idem que précédemment.

Le répertoire racine : /

Tous les répertoires du système ont un ancêtre commun. Ce répertoire s'appelle **répertoire racine**, ou *root* en anglais. On le désigne par /.



Exercice 31 (Répertoire racine) *En partant de votre répertoire personnel, remontez vers le répertoire parent, puis remontez encore... Que se passe-t-il quand vous atteignez le répertoire racine ?*

D'autres répertoires du système

Il existe bien sûr beaucoup de répertoires en dehors de votre répertoire personnel. Ce sont les répertoires contenant les fichiers qui permettent au système de fonctionner, et les fichiers correspondant aux applications que vous pouvez utiliser. Voyons-en quelques-uns :

/bin/ Contient les fichiers exécutables strictement nécessaires au fonctionnement du système.

/usr/bin/ Contient la plupart des fichiers exécutables du système. Il y en a beaucoup !

/etc/ Contient les fichiers de configuration globaux du système.

/var/ Contient les données gérées par le système.

Exercice 32 (Visite dans le terminal) *Rendez-vous dans chacun de ces répertoires et regardez les fichiers qu'il contient. Trouvez le fichier correspondant à l'exécutable `nautilus` . . .*

4.3.7 Manipulations de base sur les fichiers et répertoires

Cette partie vous permet d'apprendre à naviguer efficacement dans l'arborescence sans utiliser `nautilus`, mais simplement à partir d'un *shell*. Vous n'utiliserez `nautilus` que pour surveiller les changements d'état.

Lister les fichiers et répertoires : `ls`

Le premier besoin éprouvé est de pouvoir répondre à la question : Quels sont les fichiers que je possède ? Pour cela, on dispose d'une commande déjà vue section 4.3.2, la commande `ls` (abréviation de `l i s t`), qui, lorsqu'on la tape au clavier, demande au système d'afficher la liste des noms des fichiers que l'on possède.

Tapez donc

```
ls
```

Le système doit vous répondre par une liste de noms qui sont les noms de vos fichiers. Ce sont des fichiers que nous avons installés sur la machine pour que vous puissiez travailler.

À partir d'ici, vous ne devriez plus avoir besoin de `nautilus`. Comme expliqué ci-dessus, il est très important pour vous d'apprendre à vous servir de la ligne de commande efficacement, donnez-vous comme discipline de ne plus utiliser `nautilus`, sauf éventuellement pour vérifier que l'affichage graphique est cohérent avec ce que vous faites en ligne de commande.

Créer un répertoire : `mkdir`

Les répertoires sont un moyen très efficace de classement, encore faut-il savoir en créer. Il existe pour cela une commande qui s'appelle `mkdir`, abréviation de `m a k e d i r` ectory. On l'utilise en lui donnant comme paramètre le nom du répertoire que l'on désire créer.

Exercice 33 (Création d'un répertoire TP) Tapez par exemple :

```
mkdir TP
```

*pour créer un répertoire de nom TP. Utilisez ensuite la commande **ls** pour vérifier que vous avez bien maintenant un autre objet de nom TP et de type répertoire. Utilisez la commande **cd** pour vous positionner sur ce répertoire, et ensuite utilisez **ls** pour voir ce qu'il contient. La commande **ls** ne répond rien, indiquant ainsi que le répertoire est vide. Vérifier dans **nautilus**.*

Faites maintenant :

```
mkdir Python
```

```
ls
```

Vous venez de créer un répertoire dans lequel vous pourrez travailler vos TP de Python d'ici peu.

Détruire un fichier : **rm**

Quand on ne désire plus conserver un fichier, on peut l'effacer de l'espace disque à l'aide de la commande **rm** (abréviation de **r e m o v e**). Utilisez **ls** pour vérifier que vous possédez un fichier qui porte le nom **brouillon**. Détruisez ce fichier en tapant :

```
rm brouillon
```

utilisez ensuite **ls** pour vérifier qu'il a bien disparu. Vérifier aussi dans **nautilus**.

Pour supprimer un répertoire et tout ce qu'il contient, utilisez la commande :

```
rm -r repertoire
```

Si la commande vous demande confirmation pour chaque fichier et que ce demandes vous ennuit, utilisez l'option **-f** de la commande **\rm** (il y a un anti-slash devant **rm**), mais avec précautions...



Attention !

L'effacement est définitif!

Avec le système Unix, il n'y a aucun moyen de récupérer un fichier que l'on a détruit. Bien réfléchir avant d'utiliser **rm**.

Copier un fichier : **cp**

Il arrive souvent que l'on veuille dupliquer un fichier, pour faire une sauvegarde avant modification par exemple. On dispose d'une commande permettant de procéder à cette opération de duplication, il s'agit de la commande **cp** (abréviation de **c o p y**). Cette commande admet deux paramètres qui sont respectivement le nom du fichier original, et le nom du fichier que l'on désire créer. Faites une copie du fichier **factorielle.py** en tapant :

```
cp factorielle.py nom-de-fichier
```

en utilisant un nom de votre choix pour *nom-de-fichier*. Puis, vérifiez à l'aide de `ls` qu'un nouveau fichier vient d'apparaître, et vérifiez à l'aide de `less` [fn : : Une commande que l'on verra plus loin, mais que vous pouvez tout de même utiliser ici : elle liste le contenu d'un fichier dont le nom est donné en argument.] que son contenu est bien le même que celui de `factorielle.py`.

Pour copier un répertoire, on ajoute l'option `-r` à la ligne de commande :

```
cp -r repertoire-source destination
```

Changement du nom d'un fichier (et déplacement) : `mv`

La commande permettant de changer le nom d'un fichier est la commande `mv` (abréviation de `m o v e`) qui, comme la commande `cp` admet deux paramètres. Le premier paramètre est l'ancien nom du fichier, et le deuxième paramètre est le nouveau nom :

```
mv ancien-nom-de-fichier nouveau-nom-de-fichier
```

Exercice 34 (Changement de nom) *Changez le nom d'un fichier et vérifiez à l'aide de `ls` et `less` que tout s'est bien passé.*

La même commande permet de déplacer un fichier d'un répertoire à un autre. On l'utilise comme ceci :

```
mv ancien-repertoire/ancien-fichier \
   nouveau-repertoire/nouveau-fichier
```

ou bien

```
mv ancien-repertoire/ancien-fichier nouveau-repertoire
```

Exercice 35 (Déplacement de fichier) *Déplacez le fichier `LISEZ_MOI` dans le répertoire `stage-unix`, puis remettez-le dans le répertoire courant avec :*

```
mv stage-unix/LISEZ_MOI .
```

(le `.` désigne le répertoire courant)



A ce stade, vous devriez pouvoir sans problème résoudre l'étape B1 du jeu de piste. Pour l'étape B2, elle vous demandera un mot de passe, que voici : « `jeu2piste` » (sans les guillemets).

Chapitre 5

Applications utiles

5.1 Quelques applications déjà vues par ailleurs

- Navigateur web **Firefox** : vu dans la section 3.1, page ??.
- Lecteur de courriel **Thunderbird** : vu dans la section 3.4, page ??.
- Éditeurs de texte avancés **Atom**, **Emacs** et **gVim** : présentés plus tard dans les sections 6.1, 6.2 et 6.3.

5.2 Bureautique : LibreOffice

Vous connaissez probablement le traitement de texte **Microsoft Word** et le reste de la suite **Microsoft Office**. Ces applications sont disponibles sur les machines Windows de l'Ensimag.

La suite **LibreOffice** est une autre suite bureautique, sans doute moins connue, mais de qualité tout à fait comparable. Elle présente entre autres les avantages d'être gratuite (et libre), et de fonctionner sur beaucoup de systèmes d'exploitation, dont l'Unix que vous utilisez ici (mais aussi Windows). **LibreOffice** est basée sur une autre suite bureautique libre, appelée **OpenOffice.org**, qui est encore utilisée sur certaines machines.

Notez que cette suite vous permet d'ouvrir et d'éditer des fichiers créés avec **Microsoft Office**, comme les `.doc`, « `.xls` » ou « `.ppt` » par exemple. Le format de fichier par défaut de **LibreOffice** s'appelle **OpenDocument** (extensions `.odt` pour le traitement de texte, `.ods` pour le tableur, `.odp` pour le module présentation...).



Le saviez-vous ?

Le format OpenDocument est maintenant un standard ISO et est supporté par les dernières versions de **Microsoft Office** (même si la compatibilité n'est pas totale). Pour les versions antérieures de **Microsoft Office**, il faut installer un plugin pour les lire, ou alors utiliser les formats de Microsoft lors des sauvegardes avec **LibreOffice**.

La suite se décompose en plusieurs outils, dont :

- `oowriter` ou `lowriter` (le nom `oowriter` vient d'un ancêtre de LibreOffice : `OpenOffice.org`) pour le traitement de texte, utile pour les compte-rendus, rapports ;
- `oocalc` ou `localc` un tableur comparable à **Microsoft Excel** ;

- `oodraw` ou `lodraw` un logiciel de dessin vectoriel à l’instar d’Adobe `Illustrator`, `CorelDraw` ou `Inkscape` ;
 - `ooimpress` ou `loimpress` pour les présentations (semblable `Microsoft PowerPoint`).
- Rendez-vous sur <https://www.libreoffice.org/> pour plus d’informations.

5.3 Visualiser des images : Eye of Gnome, Gthumb...

Pour visualiser des images (fichiers JPEG, PNG, GIF...), il existe une multitude de petits programmes, au choix. On pourra citer `Eye of Gnome` (commande `eog`), ou `gthumb`. Pour visualiser une image contenue dans `mon-image.jpg`, il suffit donc d’utiliser la commande `eog mon-image.jpg`.

5.4 Visualiser des documents PDF : Evince, Acrobat Reader...

Un format très utilisé pour des documents imprimables est le format PDF, c’est même un standard ISO depuis 2008. Un fichier PDF n’est pas fait pour être édité, mais surtout pour être visualisé à l’écran, et éventuellement imprimé. Là encore, il existe une multitude de logiciels pour lire des fichiers PDF, les plus communs sont `Acrobat Reader`, le lecteur officiel d’Adobe (commande `acroread`), ou le lecteur de l’environnement `Gnome`, `evince`.

5.5 Dessin et retouche d’images : GIMP

Pour la retouche de photographies numériques ou le dessin (bitmap), les habitués de `Photoshop` retrouveront une partie de leurs habitudes avec `GIMP` (commande `gimp`).

5.6 Composition de documents : L^AT_EX

`LaTeX` (prononcer « latec ») est un logiciel comparable à un logiciel de traitement de texte, mais suivant une approche totalement différente : on édite le texte avec des commandes de mise en forme comme s’il s’agissait d’un programme, que l’on compile pour obtenir le document. Il permet d’obtenir des documents de grande qualité typographique avec un effort raisonnable.

Toutefois, `LATEX` requiert un apprentissage initial plus important que celui qui est nécessaire pour les logiciels de type `Microsoft Word` ou `LibreOffice`. Pour cette raison, il est peu connu du grand public, mais très utilisé des informaticiens, mathématiciens et physiciens devant produire un contenu parfois complexe. Il va sans dire que ce document est écrit en `LATEX` !

Le principe de `LATEX` est d’écrire un document texte (`fichier.tex`), qu’on peut compiler avec par exemple `pdflatex fichier.tex` pour obtenir un fichier `fichier.pdf`.

Inutile d’en dire plus, vous avez eu ou vous allez avoir d’ici peu une initiation à `LATEX` par vos enseignants de méthodes numériques.

Exercice 36 (Ouvrir des fichiers de types différents) *Regardez le contenu du répertoire `exemples` qui se trouve à la racine de votre compte. Essayez d'ouvrir chaque fichier qui s'y trouve (par exemple avec `libreoffice`, `gimp`, `evince`, et `less`).*

5.7 Tout ça est gratuit ?

Tous les logiciels mentionnés dans cette section, comme beaucoup de logiciels que l'on utilise fréquemment sous Linux, sont disponibles gratuitement. En fait, une grande partie de ceux-ci ne sont pas seulement gratuits : ils sont aussi **libres**, c'est-à-dire que vous pouvez non seulement les utiliser, mais aussi les distribuer, étudier leur fonctionnement et les modifier sans contrepartie¹.

Le mouvement du logiciel libre est né au début des années 1980, et aujourd'hui les logiciels libres sont pour beaucoup de très bonne qualité, et un grand nombre d'utilisateurs les préfèrent à leurs équivalents propriétaires indépendamment de leur prix. Pour plus d'informations, on pourra consulter la page suivante : <https://www.gnu.org/philosophy/free-sw.fr.html>



Ce chapitre devrait vous avoir donné les clés pour aller jusqu'à l'étape D1 du jeu de piste.

1. Ce n'est par exemple pas le cas d'`acroread`, tout comme de certains greffons (*plugins*) de LibreOffice.

Chapitre 6

Les éditeurs de texte

Avant tout, qu'est-ce que ça veut dire, un « éditeur de texte » ? Vous avez sûrement déjà utilisé un logiciel de « traitement de texte », qui permet de taper du texte, de le mettre en forme (gras, italique, taille et choix des polices...). Un éditeur de texte permet avant tout d'éditer du texte brut : pas de mise en forme, pas de dessins, juste du texte : une suite de caractères. Cela peut sembler restrictif, mais en informatique, on trouve du texte brut partout : un programme écrit dans un langage de programmation classique est un fichier texte, une page HTML est un fichier texte, la plupart des fichiers de configuration des logiciels en sont également...

On pourrait essayer d'éditer le texte d'un programme Python avec un outil de traitement de texte classique, comme `Microsoft Word` ou `LibreOffice`, mais la plupart des fonctionnalités offertes par ces programmes seraient inutiles ; à l'inverse, il manquerait énormément de fonctions spécifiques à l'édition de texte brut.

Il existe beaucoup d'éditeurs de texte sous Unix. Nous vous présentons ici `Atom`, `GNU Emacs` et `Vim`, qui sont les trois éditeurs « classiques », tous trois très puissants, mais demandant un peu d'apprentissage. Choisissez-en un des trois, et suivez en détail l'introduction qui vous est proposée.

Votre éditeur de texte est l'outil que vous allez avoir devant vous la majorité du temps quand vous programmerez. Il est très important d'être à l'aise, et surtout d'être *rapide* dans son utilisation. Ne vous arrêtez pas au minimum vital (ouvrir/enregistrer). Posez-vous la question du temps qu'il vous faut pour des tâches classiques comme : faire un rechercher-remplacer, placer le curseur sur un numéro de ligne donné, indenter un bloc de code... Si la réponse est supérieure à environ 5 secondes, vous n'utilisez probablement pas la bonne méthode (par exemple, vous ne connaissez pas le raccourci clavier), et vous allez perdre *beaucoup* de temps par la suite.

6.1 Atom

`Atom` est l'éditeur de texte recommandé à l'Ensimag. Il s'agit d'un projet relativement jeune porté par Github. `Atom` se veut à la fois moderne et adapté aux besoins des développeurs. On trouvera donc l'indentation automatique du code, la coloration syntaxique appropriée au langage de programmation, l'auto-complétion, et plein d'autres outils qui facilitent le développement.

À l'instar de ses aînés que sont `Emacs` et `vim`, `Atom` propose une navigation au clavier permettant une certaine rapidité d'écriture et « d'accélération » de certaines tâches grâce aux nombreux raccourcis proposés. Néanmoins, l'ensemble des actions sont réalisables à

la souris, ce qui pourra rassurer certains d'entre vous le temps d'apprendre à exploiter pleinement l'éditeur.

Atom a cependant le défaut d'être un éditeur entièrement graphique. Contrairement à ses aînés, il ne peut pas être exécuté directement dans un terminal. **Atom** est aussi beaucoup moins répandu. Ces deux points font qu'il vous sera probablement nécessaire de connaître quelques bases d'**Emacs** et **vim**.

Vous pouvez retrouver une introduction à **Atom** sur le Wiki « Premiers pas avec Atom et Python ».

Exercice 37 (Atom) *Suivez les explications de la page « Premiers pas avec Atom et Python ». Revenez à ce document quand vous aurez terminé.*

6.2 GNU Emacs

GNU Emacs (*aka Emacs* par la suite) a longtemps été l'éditeur de texte recommandé à l'Ensimag. Il est un peu déroutant pour les débutants, mais on s'habitue rapidement aux commandes de base. L'intérêt d'**Emacs** est qu'il a **beaucoup** de fonctionnalités¹ qui facilitent la vie du programmeur, dans la plupart des langages de programmation.

Par exemple, **Emacs** vous aide à *indenter* votre code, à trouver rapidement les erreurs de syntaxe, et il peut même taper une partie du code pour vous !

L'introduction à **Emacs** en couleur et en images se trouve sur le Wiki, « Premiers pas avec Emacs et Python ».

Exercice 38 (Emacs) *Suivez les explications de la page « Premiers pas avec Emacs et Python ». Revenez à ce document quand vous aurez terminé.*

6.3 L'éditeur Vim

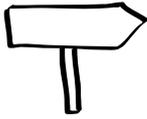
Vim est un éditeur moins intuitif, mais très puissant. Ses raccourcis claviers sont étudiés pour que les tâches courantes puissent se faire en appuyant sur un minimum de touches. Un autre avantage de **Vim** est qu'il est disponible sur pratiquement tous les systèmes de la famille Unix. Connaître les bases sous **vi** est indispensable pour un bon administrateur système, car c'est parfois le seul éditeur installé sur des machines un peu exotiques.

La version historique, **vi**, est assez rudimentaire, mais la version améliorée appelée **Vim** (pour « **vi** improved ») et sa variante graphique **gVim** sont aujourd'hui activement développées, et fournissent tout ce qu'un programmeur peut attendre d'un bon éditeur.

En bref, si vous êtes déjà informaticien chevronné et que vous cherchez un éditeur flexible et rapide, regardez ce que **Vim** peut faire pour vous. Si vous êtes débutant en informatique, à moins d'être particulièrement curieux, passez votre chemin pour l'instant, vous n'apprécierez probablement pas **Vim**.

Comme pour **Emacs**, vous trouverez une introduction plus détaillée sur le Wiki, « Vim ».

1. Ses détracteurs prétendent qu'**Emacs** est un bon système d'exploitation auquel il manque un bon éditeur de texte.

**Jeu de piste**

Vous devriez sans problème pouvoir résoudre les étapes D1 à D3. Le prochain chapitre vous donnera les clés pour résoudre un bon nombre d'étapes. Si vous avez des difficultés pour le jeu de piste, commencez par avancer dans la lecture du document, et revenez sur le jeu de piste ensuite.

Chapitre 7

Commandes et outils indispensables

7.1 Les informations associées aux fichiers et répertoires

L'organisation des fichiers sous Unix est expliquée en parallèle avec le gestionnaire de fichier graphique *nautilus* vu dans la section 4.3.2.

7.1.1 `ls` revisité

Un utilisateur gère deux types d'objets informatiques : les fichiers et les répertoires. La première des choses qu'il doit savoir faire est déterminer le type de chaque objet.

Pour cela, il faut utiliser la commande `ls` sous une forme plus élaborée que ce que nous avons fait jusqu'à présent, en utilisant l'argument `-l` (la lettre « l », pas le chiffre « 1 ») de la commande. Cet argument (`l` pour long) demande à `ls` de lister les objets en donnant plus d'informations que simplement leurs noms.

Tapez :

```
ls -l
```

et vous devez obtenir quelque chose du style de :

```
total 8
-rw-r--r-- 1 vous etudiants 404 Jul 13 19:12 LISEZ_MOI
drwxr-xr-x 5 vous etudiants 4096 Jul 13 19:12 stage-unix/
```

Vous verrez peut-être apparaître un point (.) derrière la première colonne. Vous pouvez l'ignorer pour l'instant.

Cette fois-ci, `ls` a listé les objets du répertoire courant. Pour lister les fichiers d'un répertoire en particulier, faites :

```
ls -l stage-unix
```

et vous devez obtenir quelque chose comme :

```
total 36
-rw-r--r-- 1 vous etudiants 5276 Jul 13 19:12 brouillon
drwxr-xr-x 2 vous etudiants 4096 Sep 14 2015 exemples/
-rw-r--r-- 1 vous etudiants 1689 Jul 13 19:12 factorielle.adb
```

```
-rw-r--r-- 1 vous etudiants 1319 Jul 13 19:12 factorielle.py
-rw-r--r-- 1 vous etudiants  127 Jul 13 19:12 hello.adb
-rw-r--r-- 1 vous etudiants  118 Jul 11 20:20 hello.py
lrwxrwxrwx 1 vous etudiants    6 Jul 13 19:12 lien-vers-shell -> shell//
drwxr-xr-x 2 vous etudiants 4096 Jul 13 19:14 livre/
drwxr-xr-x 3 vous etudiants 4096 Jul 13 19:12 shell/
```

`ls -l` liste les fichiers en en mettant un par ligne, le nom de l'objet se trouvant à la dernière colonne. Les autres colonnes contiennent d'autres informations que le système mémorise avec chaque objet.

- pour la première colonne, nous ne nous intéresserons ici qu'au premier caractère (voir section 7.1.4 pour le reste). Ce premier caractère est :
 - soit « - », indiquant ainsi que l'objet est un fichier normal ;
 - soit « d » indiquant ainsi que l'objet considéré est un répertoire ;
 - soit « l », indiquant ainsi que l'objet est un **lien symbolique** (cf. section 7.2.3). Vous pouvez donc voir que les objets ayant pour nom `hello.py`, `brouillon` et `factorielle.py` sont des fichiers, alors que les objets de nom `courrier` et `livre` sont des répertoires, et l'objet `lien-vers-shell` est un lien symbolique.
- la troisième colonne contient le nom du propriétaire de l'objet, ici `vous`.
- la quatrième colonne contient le nom du groupe de l'objet, ici `ens1a`.
- la cinquième colonne contient la taille de l'objet exprimée en octets, par exemple 5276 octets pour le fichier `brouillon`.
- les colonnes 6, 7 et 8 contiennent la date et l'heure de la dernière modification de l'objet, par exemple le 13 juillet de l'année courante, à 19h14 pour l'objet `livre`.

Le gestionnaire de fichiers graphique `nautilus` affiche une icône spécifique pour différencier les répertoires des fichiers. Mais ce n'est pas la seule information que l'on peut observer avec cet outil :

Exercice 39 (Les informations détaillées avec `nautilus`) *Basculez l'affichage de `nautilus`, pour le passer en mode **Liste détaillée** (icône « *View items as a list* » à droite de la barre de localisation).*



Le saviez-vous ?

`nautilus` indique aussi un **type** (image, archive...) pour le fichier. Cela n'est pas stocké par le fichier ou le répertoire, mais « deviné » en fonction de son extension ou du début de son contenu. La commande essayant de deviner le type d'un fichier est :

```
file nom_du_fichier
```

**Le saviez-vous ?**

Les « tailles » des répertoires représentent ici l'espace disque occupé par « l'objet répertoire », mais pas par les fichiers qu'il contient. Pour avoir la taille des fichiers et des sous-répertoires, on peut faire clic droit puis **Propriétés...** sur un répertoire avec **nautilus**, ou utiliser les commandes vues dans la section 7.1.6.

7.1.2 Notion de répertoire courant : `cd` et `pwd`

À tout instant, le système connaît un répertoire particulier de l'utilisateur comme étant son répertoire courant, et les commandes ont un effet relatif à ce répertoire. Par exemple la commande `ls` qui a été vue précédemment ne liste pas tous les fichiers et répertoires de l'utilisateur, mais seulement ceux qui se trouvent dans le répertoire courant (En anglais, *working directory*). On peut changer de répertoire avec la commande `cd` (pour **C**hange **D**irectory), et afficher le nom du répertoire courant avec la commande `pwd` (pour **P**rint **W**orking **D**irectory).

Exercice 40 (`cd`) Essayez de changer de répertoire avec `cd stage-unix/livre`, listez le contenu avec la commande `ls`, puis revenez.

7.1.3 Quelques répertoires particuliers**Répertoire courant et répertoire parent : `.` et `..`**

Chaque répertoire contient deux répertoires particuliers : « `.` », qui désigne le répertoire courant, et « `..` » qui désigne son répertoire parent. Ce dernier s'utilise par exemple pour revenir « un niveau de répertoire au-dessus » avec `cd ..`

Le répertoire « `.` » peut paraître inutile : « `cd .` » est une commande qui ne fait rien, « `ls ./fichier` » est équivalent à « `ls fichier` », mais il peut être utile, par exemple dans la commande `mv stage-unix/LISEZ_MOI .` que nous avons vu un peu plus haut.

Exercice 41 (`ls`) Essayez de faire `ls -a`

**Le saviez-vous ?**

La commande `ls -a` permet également d'afficher les répertoires et fichiers cachés. On distingue ces fichiers des autres par la présence d'un point (`.`) en première position dans leur nom. Par défaut, la commande `ls` ne les affiche pas.

Le répertoire « HOME » : `~`

Nous l'avons vu : sous Unix, chaque utilisateur a un répertoire personnel, ou répertoire « maison ». On peut y accéder avec le caractère `~` (lire « tilde »). Par exemple, `ls -l ~/LISEZ_MOI` donne les détails sur le fichier `LISEZ_MOI` se trouvant dans votre répertoire personnel, quel que soit le répertoire dans lequel on l'exécute.

**Le saviez-vous ?**

La commande `cd` toute seule change l'état de votre shell pour le replacer dans votre répertoire d'origine. C'est l'équivalent de `cd ~`.

Le répertoire racine : /

Le répertoire racine vu en section 4.3.6 s'appelle aussi « / » (*slash* en anglais).

Exercice 42 (Répertoires particuliers) Essayez la commande `cd ~jdpunix`. Elle vous amène dans le répertoire de l'utilisateur `jdpunix`. Faites maintenant `ls` pour en voir le contenu. Depuis ce répertoire, faites `ls ~` pour afficher le contenu du vôtre. Faites la même chose avec `/` à la place de `~jdpunix`, puis revenez chez vous avec `cd ~/`, ou plus simplement `cd`.

7.1.4 Droits des fichiers et répertoires

Chaque fichier (ou répertoire) possède un ensemble d'attributs définissant les droits d'accès à ce fichier pour chaque utilisateur du système. Chaque fichier offre des droits spécifiques à trois catégories d'utilisateurs exclusives :

1. la catégorie *user* concerne le propriétaire principal du fichier.
2. la catégorie *group* concerne un ensemble d'utilisateurs ayant donc des droits particuliers sur le fichier. Chaque utilisateur appartient à un ou plusieurs groupes. C'est l'administrateur système qui gère ces groupes.
3. la catégorie *others* concerne tous les autres utilisateurs.

Lorsqu'un utilisateur est susceptible d'appartenir à plusieurs catégories, le système considère en fait qu'il appartient à la plus prioritaire : *user* > *group* > *others*. A quel groupe vos fichiers appartiennent-ils en majorité ?

Les droits de chacune de ces catégories d'utilisateurs sur un fichier sont définis par l'autorisation ou l'interdiction de chacun des trois types d'accès suivant : lecture (*Read* en anglais), écriture (*Write*), et exécution (*eXecute*).

**Le saviez-vous ?**

Pour les répertoires, *Read* correspond au droit de lister le contenu (commande `ls`); *Write* correspond au droit de créer ou détruire un objet dans le répertoire (commande `rm`); et *eXecute* correspond au droit de traverser le répertoire (commande `cd`).

Chaque objet possède donc neuf attributs pour déterminer les droits d'accès de chaque utilisateur. Les attributs sont listés avec l'aide de la commande `ls -l` vu précédemment (rappel : le premier caractère est le type d'objet cf. section 7.1.1);

```
-rwxr-xr-x 3 cassagne ensila 3544 Sep 15 13 :27 brouillon
```

user
 group
 other

Les attributs sont regroupés par catégorie d'utilisateurs : sur ces neuf caractères, les trois premiers concernent les droits *user*, les trois suivants les droits *group*, et les trois derniers les droits *others*. Lorsque l'accès est interdit, le caractère correspondant est un tiret « - ». Lorsque l'accès est autorisé, le caractère correspondant est **r**, **w** ou **x** suivant le type d'accès.

Par exemple, `drwxr-xr--` indique que le fichier correspondant est un répertoire qui autorise à son propriétaire tous les types d'accès, interdit aux utilisateurs du groupe de créer ou détruire le contenu du répertoire, et autorise aux autres uniquement de lister le contenu sans y accéder avec la commande `cd`.

Exercice 43 (Droits) *Essayez de trouver des répertoires qui vous sont interdits (par exemple chez votre voisin) ou de renommer des fichiers qui ne vous appartiennent pas. Quel est le message d'erreur ?*

7.1.5 Modification des droits d'accès : `chmod`

La commande pour modifier les droits d'accès des fichiers est :

```
chmod permissions liste_d_objets
```

Les droits sont désignés par une liste d'expressions séparées par des virgules. Ces expressions indiquent :

1. les catégories d'utilisateurs concernées par le changement :
 - **u** pour *user* ;
 - **g** pour *group* ;
 - **o** pour *others* ;
 - et **a** pour toutes
2. suivies de l'opération de changement :
 - soit **+** pour autoriser ;
 - soit **-** pour interdire.
3. et finalement les types d'accès concernés (**r**, **w** et **x**).

Par exemple, la ligne suivante autorise le propriétaire et le groupe en écriture et exécution, et interdit le groupe et les autres en lecture, sur le fichier `LISEZ_MOI`.

```
chmod ug+wx,go-r LISEZ_MOI
```

Ainsi, si `LISEZ_MOI` était en mode `rw-r--r--` avant l'exécution de cette commande, il est en mode `rw-x-wx--` après.

Exercice 44 (Droits) *Faites des expériences pour répondre aux questions suivantes :*

1. *Pouvez-vous modifier un fichier sur lequel vous avez les droits en écriture, mais qui se trouve dans un répertoire dans lequel vous n'avez pas le droit d'écrire ?*
2. *À quelles conditions pouvez-vous lire dans un fichier qui se trouve dans un répertoire interdit à la lecture pour vous ? Et dans un répertoire interdit en exécution ?*
3. *Que se passe-t-il si vous interdisez en exécution le répertoire courant ? Peut-être ne vaut-il mieux pas essayer sur son répertoire d'origine, pourquoi est-ce dangereux ?*



Le saviez-vous ?

ACL : Access Control List

Il est possible de positionner beaucoup plus finement les droits d'accès afin de ne les donner par exemple qu'à un utilisateur en particulier. Cela est réalisé avec les commandes `getfacl` et `setfacl`. Par exemple, pour donner le droit à l'utilisateur cassagne de lire le fichier `toto.txt` :

```
setfacl -m u:cassagne:r toto.txt
```

Pour plus d'informations, rendez-vous sur le Wiki, « ACL ».

7.1.6 Utilisation de l'espace disque : `du` et `df`

La commande `du` (pour `d isk u sed`) permet de connaître l'utilisation d'espace disque d'un fichier ou d'un répertoire. Sans argument, elle donne l'utilisation du répertoire courant, mais peut aussi prendre un autre répertoire comme argument. En général, on l'utilise avec l'option `-s` (`s ummary`) et `-h` (`h uman readable`).

La commande `df` (pour `d isk f gree`) donne l'espace disque pour chaque disque dur de la machine (en fait, chaque partition).

Exercice 45 (Espace disque) *Utilisez la commande `du -sh` dans votre répertoire personnel. Comparez avec `df -h`.*

7.2 Manipulation de fichiers : contenu, organisation

7.2.1 Visualisation du contenu : commandes `more` et `less`

Pour voir le contenu d'un fichier, le plus efficace est d'utiliser la commande `less` qui admet en paramètre le nom du fichier concerné (sur certains systèmes, la commande `less` n'est pas disponible et il faut se contenter de l'antique `more`, qui est à peu près la même, sauf qu'elle en fait moins!).

En utilisant la commande `ls`, vous avez vu que vous disposez d'un fichier dont le nom est `factorielle.py`, vous allez regarder son contenu. Pour cela, tapez :

```
less factorielle.py
```

Sur votre écran doit s'afficher le début du fichier. Quand on est sous le contrôle de `less`, l'utilisateur peut émettre plusieurs ordres, dont les principaux sont :

-  , pour faire apparaître une ligne supplémentaire du fichier.
-  , (c'est-à-dire appui sur la barre d'espace), pour faire apparaître une page d'écran supplémentaire du fichier.
-  et  pour descendre et remonter dans le texte (les deux touches sous l'index et le majeur de votre main droite dans la position de repos).
-  pour arrêter l'exécution de `less`.
-  pour rechercher un mot dans le texte.
-  pour n'afficher que les lignes contenant un motif de texte.
-  pour avoir de l'aide.

Exercice 46 (less) *Utilisez ces ordres pour constater leur effet et regarder le contenu du fichier `chap1` (toujours dans le répertoire `~/stage-unix/livre`).*

Exercice 47 (Page par page) *Une autre utilisation courante de la commande `less` est de regarder la sortie d'une commande page par page. Essayez la commande `ls /usr/bin/`, puis `ls /usr/bin/ | less` (le caractère `|` se lit « pipe », et s'obtient avec  +  sur un clavier azerty. Il sera expliqué en détail à la section 7.9).*

7.2.2 Astuce : Utilisation efficace du terminal

Au point où on en est, le lecteur a le droit de se demander comment être efficace avec un interpréteur de commandes : les commandes sont parfois longues, et s'il faut les taper en entier, une interface graphique serait plus rapide. La réponse est, bien entendu, qu'on ne tape pas **tout** !

Voici quelques exemples :



Le saviez-vous ?

L'historique

On peut reprendre une commande déjà exécutée en remontant dans l'historique avec la touche  (et ).



Le saviez-vous ?

Complétion de commande

On peut taper le début du nom d'une commande et appuyer sur la touche tabulation, ou  (au-dessus de ) : l'interpréteur complète tout seul. Lorsqu'il y a plusieurs possibilités, on peut appuyer une seconde fois pour obtenir la liste des possibilités.



Le saviez-vous ?

Complétion de noms de fichiers

De la même manière, lorsqu'une commande prend en argument un nom de fichier, on peut entrer le début du nom et appuyer sur pour compléter.



Le saviez-vous ?

Édition de ligne

Les commandes d'éditations de base sont disponibles dans l'interpréteur de commande : et pour se déplacer, et pour effacer... combiné avec l'historique, on peut exécuter une série de commandes similaires rapidement.

Exercice 48 (Tab) Depuis votre répertoire personnel, entrez *fire*, puis appuyez sur . Effacez la ligne (en faisant + , ou avec la touche , puis entrez *less LIS* et appuyez sur .

7.2.3 Notion de lien symbolique

Un lien symbolique est une référence à un fichier ou répertoire. Pour créer un lien symbolique, on utilise :

```
ln -s chemin_pointé nom_du_lien
```

Par exemple, pour créer un lien vers son `~`, sous le répertoire courant, on ferait :

```
ln -s ~/ chezmoi
```

On peut utiliser `ls -l` pour identifier la cible d'un lien symbolique cf. section 7.1.1.

La plupart des opérations (ouverture, lecture, écriture) sur un lien symbolique le dérèrent automatiquement et opèrent sur sa destination. La suppression (`rm`) et le déplacement/renommage (`mv`) portent sur le lien lui-même et n'affectent pas la cible.

La commande `rm chezmoi` effacera donc le lien et non pas votre home !



Le saviez-vous ?

Les liens symboliques sont connus sous le nom de « raccourcis » (`.lnk`) sous Windows et « alias » sous Mac OS.

7.2.4 Recherche de fichier : `find` et `grep`

Trouver un fichier : `find`

La commande `find` est une bonne illustration de la différence d'expressivité entre les interfaces graphiques traditionnelles et une interface de type textuelle.

Cette commande permet, comme son nom l'indique, de trouver un fichier. Il existe de très nombreux critères possibles. La liste des critères et des tests est disponible dans la page de manuel (pour apprendre à se servir du manuel : cf. section 7.5 ,??).

La syntaxe de la commande est de la forme :

```
find Répertoire_de_départ Critères_de_sélection
```

Par exemple, pour trouver le fichier nommé `toto.txt` dans le répertoire courant ou un de ses sous-répertoires, on fait :

```
find . -name toto.txt
```

Qu'il faut lire « cherche (`find`) dans le répertoire courant (`.`) et ses sous répertoires un fichier (ou répertoire) nommé (`-name`) `toto.txt` ». On peut aussi demander la liste des fichiers se terminant d'une certaine manière :

```
find . -name '*.py'
```

« `*` » veut dire « n'importe quelle chaîne » dans ce contexte. N'oubliez pas les guillemets autour de `*.py` ou vous pourriez avoir des surprises.

On peut aussi retrouver, par exemple, les fichiers que l'on a édité il y a 6 jours dans le répertoire courant par la commande :

```
find . -mtime 6
```

Exercice 49 (Édition précédente) *Essayez de retrouver tous les fichiers que vous avez édité, y compris implicitement, lors de la précédente séance.*



Le saviez-vous ?

Utilisation avancée de find

Une des différences entre `find` et les outils graphiques, c'est la simplicité avec laquelle on peut exprimer la combinaison de ces critères : les grouper avec « `()` », faire un ET logique avec `-a` ou un OU logique avec `-o`, ou encore faire la négation `\!` d'un critère. On peut aussi indiquer une valeur inférieure ou supérieure à une quantité en ajoutant `+` ou `-`.

Par exemple, on peut retrouver tous les fichiers Python (source et bytecode) que l'on a édité pendant la semaine dernière (il y a plus de 6 fois 24 heures et moins de 14 fois 24 heures) sauf ceux que l'on a modifié après le fichier `toto.py` avec la commande suivante.

```
find . \( -name "*.py" -o -name "*.pyc" \) \
    -ctime +6 -ctime -14 \! -newer toto.py
```

Le caractère « `\` » en fin de ligne est utilisé pour indiquer que la commande continue sur la deuxième ligne. Comme vous l'avez remarqué, le ET (`-a`) n'apparaît pas dans les exemples car c'est l'opérateur par défaut.

Rechercher une chaîne dans un fichier : `grep`

La commande `grep` permet de chercher une chaîne de caractères dans un ou plusieurs fichiers. On l'utilise comme ceci :

```
grep chaine-a-rechercher fichier
```

(avec éventuellement plusieurs fichiers à la place de *fichier*).

Exercice 50 (Utilisation de `grep`) *Dans un terminal, placez-vous dans le répertoire `stage-unix/livre` (avec `cd stage-unix/livre`, ou simplement `cd livre` si vous étiez déjà dans `stage-unix`). Essayez d'entrer la commande :*

```
grep homme chap1 chap2 chap3
```

Ce n'est pas assez joli ? Essayez ceci :

```
grep --color homme chap1 chap2 chap3
```

En fait, la commande `grep` est aussi très utile pour « filtrer » la sortie d'une autre commande. Par exemple, si on fait (nous verrons un peu plus loin que cette commande liste tous les programmes tournant sur la machine) :

```
ps -A
```

La sortie de la commande est assez longue (**très** longue si on l'exécute sur un serveur comme `pcserveur.ensimag.fr` lorsqu'il est utilisé par beaucoup d'étudiants en même temps). Nous avons vu qu'il était possible de la voir page par page avec `ps -A | less`. Sur le même principe, si nous ne voulons voir par exemple que les lignes qui parlent de `Gnome`, on peut faire :

```
ps -A | grep gnome
```



Le saviez-vous ?

Expressions régulières

En fait, `grep` ne cherche pas vraiment des chaînes, mais des expressions appelées « expressions régulières ». Par exemple :

```
grep 'ab*c'
```

recherche un `a` suivi d'un nombre quelconque de `b`, suivi d'un `c`. Vous en apprendrez beaucoup plus en cours de théorie des langages.

7.3 Compression et archivage

7.3.1 Compression avec `gzip`

Les fichiers compressés utilisent moins d'espace disque et se téléchargent plus rapidement. Vous pouvez compresser un fichier à l'aide d'un outil de compression.



Attention !

Format de fichiers et communications.

Pensez à vérifier si l'outil pour décompresser est disponible avant d'envoyer un fichier compressé à quelqu'un.

Le logiciel de compression standard sous Unix est `gzip`. La commande `gzip` crée un fichier compressé et la commande `gunzip` décompresse un fichier compressé. Par convention, les fichiers compressés avec cette méthode se voient attribuer l'extension `.gz`. Par exemple :

```
gzip nom_du_fichier
```

Le fichier sera compressé et sauvegardé comme `nom_du_fichier.gz`. L'opération inverse est effectuée avec :

```
gunzip nom_du_fichier.gz
```

7.3.2 Archivage avec tar

Une archive `tar` regroupe plusieurs fichiers ou répertoires dans un seul fichier. Une archive `tar` n'est pas compressée et peut être vue comme une concaténation de fichiers. Il s'agit d'une bonne manière pour créer des sauvegardes. Généralement, le nom d'une archive `tar` se termine par l'extension `.tar`.

Pour créer une archive `tar` qui contient tout un répertoire, tapez :

```
tar -cvf nom_du_fichier.tar repertoire
```



Attention !

L'argument qui suit immédiatement l'option `f` est considéré comme l'archive par `tar`. Par exemple, si vous utilisez la commande (dangereuse)

```
tar -cvf tp-algo.py fact.py
```

`tar` va créer une archive appelée `tp-algo.py` contenant le fichier `fact.py`. Si le fichier `tp-algo.py` existait déjà, **il est écrasé!** En un mot, oublier le nom de l'archive dans la commande `tar` peut vous faire perdre votre TP.

Pour extraire le contenu d'un fichier `tar`, entrez :

```
tar -xvf nom_du_fichier.tar
```

7.3.3 Archivage puis compression avec tar

La commande `tar` ne compresse pas automatiquement les fichiers. Une archive `tar` peut être compressée avec `gzip` et dans ce cas on lui attribuera généralement une extension `.tar.gz` ou `.tgz`.

Vous pouvez effectuer un archivage et une compression dans la même opération avec :

```
tar -czvf nom_du_fichier.tar.gz repertoire
```

Et pour décompresser et extraire en même temps :

```
tar -xzvf nom_du_fichier.tar.gz
```

7.3.4 Archivage et compression avec zip

Les fichiers `.tar.gz` sont la tradition sous Unix. Ils sont également lisibles sous Windows avec des logiciels comme WinZip, mais si vous échangez des fichiers avec des utilisateurs non-Unix, vous pouvez utiliser le format `zip` pour ne pas perturber leurs habitudes. Pour archiver et comprimer à l'aide de `zip`, entrez ceci :

```
zip -r nom_du_fichier.zip repertoire
```

Pour décompresser et extraire le contenu d'un fichier `zip`, entrez :

```
unzip nom_du_fichier.zip
```

7.4 Impression de documents : `a2ps`, `cancel` et `lpstat`

Il est souvent utile d'imprimer des documents électroniques sur papier. L'Ensimag dispose de plusieurs imprimantes (plus précisément, de copieur-scanner-imprimantes). Le système utilise une file d'attente partagée : vous ne voyez qu'une imprimante (Toshiba). Quand vous lancez l'impression, rien ne se passe : vous devez vous rendre sur un copieur (celui de votre choix), vous identifier et lancer l'impression.

Le code à entrer pour vous identifier est disponible en bas de la page <https://intranet.ensimag.fr/passwords/>. Les instructions détaillées sont affichées au-dessus de chaque copieur.

Pour plus d'information, consulter la page « Système de gestion des impressions » de l'intranet de l'Ensimag.

Si vous lancez une impression et que vous réalisez que vous vous êtes trompés, vous n'avez rien à faire : la demande d'impression sera supprimée automatiquement si vous n'allez pas la chercher.



Le saviez-vous ?

Le système est très spécifique à l'Ensimag. Traditionnellement, sous Unix, on choisit son imprimante au moment de lancer l'impression, et on peut utiliser les commandes `lpstat -p nom-de-l'imprimante` pour voir les travaux en attente, `cancel numéro-du-job` pour annuler s'il n'est pas trop tard.

Vous pouvez également utiliser les fonctions « copieur » et « scanner » : suivez les instructions affichées au-dessus des copieurs.

Attention, vous avez un quota d'impression, ne gaspillez pas !

7.4.1 Imprimer un document

La plupart des logiciels graphiques auxquels vous demanderez une impression discutent directement avec le serveur d'impression. Lancer une impression est donc relativement simple.

Mais il existe aussi des programmes en ligne de commande pour faire la même chose pour tous les autres cas.

Une commande simple est :

```
a2ps nom-du-fichier
```

Elle s'occupe de convertir les formats de fichiers pour les images, ou bien de mettre en forme les documents textuels comme les codes sources par exemple. Par défaut, elle envoie le résultat à l'imprimante.

Exercice 51 (Impression dans un fichier) *Il est aussi possible d'envoyer le résultat de `a2ps` dans un fichier (donc sans l'imprimer), en utilisant l'option `-o` : `a2ps -o toto.ps moncode.py`. Essayez de le faire avec votre code Python avec plusieurs vignettes par pages en ajoutant le nombre de vignettes en option entre `-1` et `-9`.*



Le saviez-vous ?

Les commandes standards

Les commandes standards d'impression sont `lpr` et `lp`. Cependant, elles n'impriment que les fichiers PostScript, d'où l'avantage d'utiliser la commande `a2ps`, qui s'occupe pour vous des conversions et mises en forme.



Le saviez-vous ?

Format d'impression

Sous Unix, la situation est relativement simple : toutes les impressions sont faites en PostScript. Pour pouvoir imprimer un document il suffit donc d'être capable de le convertir en PostScript.



Le saviez-vous ?

Conversion entre PostScript et PDF

PostScript est l'ancêtre de PDF. Ils sont structurellement proches. PDF est optimisé pour des manipulations automatiques alors que PostScript est un langage de programmation compréhensible et manipulable par un humain. Comme la visualisation de documents PDF est souvent plus facile dans d'autres systèmes, il est souvent intéressant, et facile, de convertir d'un format dans l'autre avec les commandes `ps2pdf` et `pdf2ps`.

Comme toutes les impressions utilisent le format PostScript, il existe de nombreux outils capables de manipuler ces fichiers pour changer la présentation, faire des livres (comme ceci !) au format A5, extraire des pages, ou une partie d'une page, etc.

7.4.2 Plus d'information

Vous trouverez plus de détails sur le fonctionnement de l'impression, les noms des imprimantes, sur EnsiWiki, « Tout pour imprimer à l'Ensimag ».

7.5 Obtenir de l'aide : `--help` et `man` et `info`

Au point où nous en sommes, vous avez déjà vu une bonne dizaine de commandes. À vrai dire, il en existe en fait plusieurs milliers sur la plupart des machines Unix. Mais alors, faut-il tout apprendre par cœur ? Heureusement, non !

Pour bénéficier d'une aide rapide, vous pouvez tout d'abord taper le nom de votre commande suivi de `--help` comme, par exemple, `mkdir --help`

Il existe également un système de documentation un peu rebutant au départ, mais assez bien fait : le `man`, ou « manuel ». Le style est en général assez technique, les « pages de `man` » sont rarement le moyen le plus simple de se familiariser avec une commande complexe, par contre, c'est souvent **très** efficace comme référence, ou pour chercher une information précise quand on sait ce qu'on veut.

L'utilisation usuelle est simplement `man commande`, où *commande* est la commande pour laquelle vous voulez obtenir de l'aide. Par exemple, pour obtenir de l'aide sur la commande `ls`, tapez `man ls`. Vous devez voir un texte d'aide, et ce texte est en fait affiché avec la commande `less`, donc vous pouvez naviguer dans ce texte comme nous l'avons vu en section 7.2.1. En particulier, la touche  permet de rechercher une chaîne de caractère rapidement dans la page.

Exercice 52 (Couleurs dans `ls`) Dans la page de `man` de `ls` (`man ls`), trouvez la partie qui décrit l'option permettant d'avoir la couleur (tapez `/color` ).

`man` permet en fait beaucoup d'autres choses. Mais pour connaître les possibilités de `man`, tout est dans `man man`, que tout bon débutant sous Unix devrait lire !

Il existe un autre système de documentation sous Unix, parfois plus convivial que `man` : la commande `info`. Tout comme le `man`, vous pouvez apprendre tout ce dont vous avez besoin avec `info info`, notamment un excellent petit tutoriel.

Les commandes comme `cd`, qui changent l'état interne de votre shell, sont décrites dans la page de man du shell (`man bash`). Mais il est également possible d'obtenir uniquement la documentation de la commande en tapant `help cd`.

7.6 Que se passe-t-il sur le serveur ?

Vous travaillez sur un serveur partagé par tous les étudiants de votre promotion. Commençons par regarder qui est en train d'exécuter des programmes sur le même serveur que vous : c'est la commande `users`.

Pour savoir qui utilise le plus de ressources sur la machine, essayez la commande `top`. Elle liste les processus (un processus est un programme en train de s'exécuter) par ordre décroissant d'utilisation du processeur.

Enfin, pour avoir tous les détails sur tous les processus qui s'exécutent sur la machine, on peut utiliser la commande `ps`.

Exercice 53 (ps) Essayez la commande `ps -u votre-login`. Elle affiche la liste des processus lancés par votre utilisateur Unix. Faites `ps u -u votre-login` : la liste est la même, mais plus joliment présentée. Terminez par `ps -A` pour avoir tous les processus du système.



Le saviez-vous ?

`lsuf` :

Il est possible d'avoir aussi la liste complète des fichiers ouverts vous appartenant avec la commande `lsuf`.

7.6.1 Notion de programme

Un programme informatique est un fichier contenant des ordres qui indiquent à l'ordinateur ce qu'il doit faire. Généralement, il se présente sous la forme d'une ou plusieurs séquences d'instructions. Un ordinateur sans programme ne fait absolument rien. En fait, c'est la capacité à suivre ces ordres qui distingue un ordinateur d'une simple machine à calculer.

7.6.2 Notion de processus

En toute généralité, un processus est un programme en cours d'exécution. Sous Unix, du point de vue de l'utilisateur, les processus semblent s'exécuter en parallèle. En réalité, comme il y a souvent un nombre plus petit de processeurs que de processus, le système d'exploitation permet à chacun des processus de s'exécuter pendant un court laps de temps, à tour de rôle. En principe, sauf surcharge de la machine, cet entrelacement est suffisamment rapide et bien ordonné pour que l'utilisateur ne le perçoive pas. C'est pourquoi lorsque beaucoup de programmes utilisant des ressources importantes (comme `Firefox`) sont lancés en même temps, l'exécution de chacun de ces programmes est ralentie.

Chaque processus a son propre espace de données et état d'exécution, qui le rend indépendant des autres.

7.6.3 La différence entre un processus et une fenêtre

Pour illustrer cet aspect, on va réaliser l'expérience suivante, qui permettra aussi de distinguer entre « processus » et « fenêtres » :

Exercice 54 (Fenêtre versus processus) 1. Affichez la liste de vos processus, par la commande `ps` :

`ps`

2. Lancer *Emacs* depuis l'interpréteur de commande :

`emacs &`

On expliquera le rôle du `&` ultérieurement à la section 8.3. Remarquez le numéro écrit dans le terminal, après le nombre entre crochet. C'est un numéro (unique) qui sert à identifier le processus : c'est son PID (processus identifier).

3. Observez de nouveau la liste de vos processus. Il y en a un de plus que précédemment : le `emacs` qui vient d'être lancé, repérable à son PID justement.
4. Tapez un court texte dans *Emacs* et le sauvez dans un fichier `toto1`.
5. Ouvrir une nouvelle fenêtre (New Frame dans le menu).
6. Observer la liste des processus (avec `ps`) : il n'y a pas de nouveau processus.
7. Dans la nouvelle fenêtre, créer un nouveau fichier `toto2` (Visit New File dans le menu File) et taper du texte, puis sauvez.
8. Observez la liste des tampons dans une des deux fenêtres (menu Buffers) (`toto1` et `toto2` sont tous les 2 présents). Essayez d'ouvrir `toto1` dans la fenêtre de `toto2`, que se passe-t-il ?
9. Lancez de nouveau `emacs` depuis l'interpréteur de commande.
10. Observez la liste des processus : il y a un nouveau processus `emacs`.
11. Tapez et sauvez un texte `toto3` dans ce dernier `emacs`.
12. Revenez sur la première fenêtre de `toto1` : `toto3` est-il présent dans la liste des fenêtres ? Pourquoi ? Ouvrir le fichier `toto3` depuis cette première fenêtre et le modifier : les modifications sont-elles prises en compte dans la dernière fenêtre ?
13. Sans sauvez le fichier `toto3` dans la première fenêtre, modifiez `toto3` dans la dernière fenêtre. Essayez de sauvez le fichier dans la première fenêtre. Que se passe-t-il ? Essayez de sauvez le fichier dans la dernière fenêtre. Que se passe-t-il ?

Expliquons maintenant ces comportements. Dans le premier cas, les 2 fenêtres sont gérées par un même processus. Celui-ci connaît donc les informations de chacune des 2 fenêtres. Dans le deuxième cas, on a lancé un nouveau processus. Son espace de données est disjoint du processus précédent : les exécutions de ces 2 processus sont indépendantes. En particulier, chacun des processus ignore qu'un autre processus est en train de modifier le même fichier ; ce n'est qu'au moment où l'autre processus sauvegarde le fichier sur le disque qu'il peut éventuellement s'en apercevoir. Si des modifications avaient été apportées au fichier dans les 2 processus, celles d'un des 2 processus vont être à refaire. Afin

d'éviter des problèmes de ce genre, pour les applications (Firefox, votre éditeur favori, etc) qui offrent la possibilité d'ouvrir plusieurs fenêtres, il est recommandé d'utiliser cette possibilité plutôt que de lancer plusieurs processus.

Enfin, tout processus appartient à un utilisateur (appelé le propriétaire). Les droits d'un processus sur les fichiers sont ceux de son propriétaire. En fait, tout programme qui s'exécute est un processus. Ainsi, la vérification des droits sur les fichiers, par exemple lors de l'exécution d'une commande `source{rm}`, se fait au regard du propriétaire du processus correspondant.

7.7 Arrêter des processus, plus ou moins gentiment

En général, pour arrêter un programme, le programme lui-même fournit une fonctionnalité « quitter », dans un menu, ou bien en cliquant sur la case « fermeture » de la fenêtre graphique en haut à droite.

Mais il arrive qu'on veuille forcer un programme à se terminer. Sous Windows, on le ferait avec le gestionnaire des tâches (accessible par exemple en faisant **Control** + **Alt** + **Suppr**). Voici plusieurs solutions sous Unix :

La première est d'utiliser la commande `kill`. Cette commande prend en argument un numéro de processus, donc, on l'utilise en général après avoir lancé la commande `ps`.

Exercice 55 (Utilisation de `kill`) *Dans un terminal, lancez la commande*

```
less LISEZ_MOI
```

Ouvrez un deuxième terminal, dans lequel vous lancez la commande `ps -u votre-login` pour afficher tous les processus que vous avez lancé. Vous devriez trouver le processus `less` que nous venons de lancer.

Faites maintenant `kill pid`, où `pid` est le numéro du processus `less` (on peut utiliser la souris pour copier-coller ce numéro). Vous devriez voir `less` se terminer et dans l'autre terminal.

En fait, la commande `kill` « demande » au processus de se terminer, mais le processus à qui on envoie ce signal peut l'ignorer. Dans ce cas, on peut utiliser la commande `kill -9`, qui tue le processus sans sommation.

Exercice 56 (Forcer un processus à se terminer) *Refaites la manipulation précédente, en utilisant `kill -9` à la place de `kill`. Comparez le message donné par `less` avec le précédent.*



Attention !

```
kill -9
```

Cette commande ferme instantanément l'application lancée sans possibilité pour elle de sauvegarder le travail en cours.



Le saviez-vous ?

Tuer tous les processus.

Pour tuer tous les processus de l'utilisateur courant, on peut faire `kill -9 -1`. En particulier, elle tue votre gestionnaire de fenêtre : c'est donc un moyen efficace mais très violent de terminer sa session ! (aucune sauvegarde possible du travail en cours)



Le saviez-vous ?

Tuer un processus uniquement à partir de son nom.

La commande `pkill nom-du-programme` est une variante de `kill` où, en paramètre, on précise non pas le pid du processus mais son nom.

Une deuxième solution, plus pratique lorsque l'application a été lancée dans un terminal, est d'appuyer sur `Ctrl` + `C`. Le terminal envoie alors un signal de terminaison au processus courant. C'est équivalent à faire `kill`.

Exercice 57 (Tuer une application avec Ctrl+c) *Faites `ls` pour vérifier l'existence dans le répertoire courant d'un fichier ayant pour nom `boucle`. Si il n'est pas dans le répertoire courant, vous pouvez le trouver avec `find . -name boucle`. Ce fichier contient un programme dont l'exécution ne finit jamais. Lancez-le en tapant :*

`boucle`

après quelques secondes d'exécution, tapez `Ctrl` + `C` et vérifiez que le programme a bien été interrompu.

Une dernière solution, pour tuer une application graphique : la commande `xkill`. On la lance depuis un terminal, puis on clique sur une fenêtre pour tuer l'application à laquelle appartient cette fenêtre.



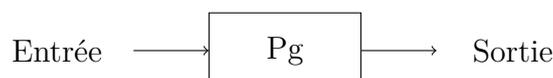
Le saviez-vous ?

Serveurs autonettoyants

Régulièrement, vers 7h du matin, la plupart des serveurs de l'Ensimag sont redémarrés automatiquement. les processus utilisateurs qui pourraient encore s'exécuter sont tués.

7.8 Redirection des Entrées Sorties

Un processus peut effectuer une transformation sur de l'information : il lit de l'information en entrée, effectue un traitement, et donne de l'information en sortie. Schématiquement, nous pouvons représenter les choses de la manière suivante :



Le processus lit l'information sur l'**entrée standard** (*standard input*, ou *stdin* en Anglais), écrit de l'information sur le **sortie standard** (*standard output* ou *stdout* en Anglais), sachant que l'entrée standard est généralement affectée au clavier, et la sortie standard est généralement affectée à l'écran.

Il existe cependant de nombreuses situations où on éprouve le besoin de faire travailler un processus en lui faisant lire l'information non pas à partir du clavier mais à partir d'un fichier, et/ou en lui faisant écrire l'information en sortie non pas sur l'écran mais dans un fichier.

Quand on réalise une telle opération, on dit que l'on a redirigé l'entrée et/ou la sortie.

Le shell permet de réaliser cette redirection de manière très simple. Quand on désire exécuter la commande *commande* en redirigeant son entrée sur le fichier *nom-de-fichier-entrée*, il suffit de taper :

```
commande < nom-de-fichier-entrée
```

Quand on désire exécuter la commande *commande* en redirigeant sa sortie sur le fichier *nom-de-fichier-sortie*, il suffit de taper :

```
commande > nom-de-fichier-sortie
```

Si on désire rediriger l'entrée et la sortie, il suffit de taper :

```
commande < nom-de-fichier-entrée > nom-de-fichier-sortie
```

Exercice 58 (Redirection de la sortie) *Pour cet exercice, nous allons utiliser une nouvelle commande, la commande **cat**, qui admet en paramètre un ensemble de noms de fichiers, et qui a pour effet de sortir sur la sortie standard, la concaténation du contenu des fichiers.*

*Revenez dans le répertoire **shell**, et regardez par **less** (ou **more**) le contenu des fichiers **m1** et **m2**, puis exécutez la commande **cat m1 m2** pour bien comprendre le fonctionnement de **cat**.*

*Maintenant, réexécutez **cat** de façon à mettre la concaténation des contenus de **m1** et **m2** dans un fichier de nom quelconque, par exemple **resu1**, et vérifiez en suite à l'aide de **more**, que le fichier **resu1** contient bien le résultat espéré.*

Exercice 59 (sort) *Utilisez la commande **sort** et la redirection d'entrée/sortie pour trier un fichier.*



Le saviez-vous ?

Faites des expériences pour répondre à la question suivante :

Lorsqu'on redirige la sortie d'un processus dans un nouveau fichier, quand le fichier est-il créé : avant, pendant ou après l'exécution du processus ?

Exercice 60 (Redirection de l'entrée) Pour cet exercice, nous allons utiliser une commande de comptage. Cette commande a pour nom `wc` (pour *w ord c out*), et est capable de compter non seulement les mots, mais également les caractères et les lignes du texte qui lui est fourni en entrée standard :

- `wc -c` : compte les caractères ;
- `wc -w` : compte les mots ;
- `wc -l` : compte les lignes.

On va d'abord utiliser cette commande sans rediriger son entrée. Tapez :

```
wc -l
```

A ce moment, le processus de comptage est en attente sur son entrée standard, c'est-à-dire le clavier, tapez donc quelques mots, et terminez par `Ctrl` + `D` juste après un `entrée` pour indiquer la fin des données.

Après cela, `wc` doit vous indiquer le nombre de mots que vous avez tapés. Utilisé de cette manière, `wc` ne sert à rien, mais on peut l'utiliser, en redirigeant son entrée, pour compter le nombre de mots (par exemple) contenu dans un fichier.

Utilisez `wc -w` pour compter le nombre de mots du fichier `chap1` qui se trouve dans votre répertoire d'origine, sans quitter le répertoire `shell`.

Exercice 61 (Redirection de l'entrée et de la sortie) Comptez le nombre de mots du fichier `chap1`, en envoyant le résultat dans le fichier `resu2`. Vérifiez à l'aide de `more` que `resu2` contient bien le résultat espéré.

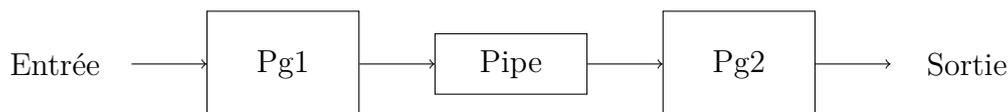
7.9 Les tuyaux, ou « pipe »

Nous avons vu dans la section 7.2.1, l'utilisation du symbole `|` dans le shell. Nous allons expliquer l'utilisation de ce symbole maintenant.

Il arrive très souvent que l'on désire faire coopérer deux processus dans une chaîne, c'est-à-dire, de manière à ce que le second effectue son travail sur l'information produite par le premier. Le schéma suivant illustre cette situation :



Pour réaliser cela, on utilise une espèce de tuyau à information entre les deux processus. Dans le jargon Unix, c'est un *pipe*.



Dans le shell, on utilise la syntaxe suivante pour demander l'établissement d'un pipe :

```
commande1 | commande2
```

Exercice 62 (Listage du contenu d'un répertoire, trié par taille) La commande `sort -n -k 5` prend un tableau en entrée et exécute le tri de ce tableau selon la valeur croissante de la 5-ème colonne. Cela dit, utilisez ce que vous avez appris dans la section 7.1.1 pour afficher le contenu d'un répertoire, trié par taille.

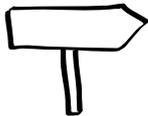
Exercice 63 (Pipe et grep) La commande `ps` peut prendre l'option `-ef` pour voir la totalité des processus tournant sur une machine. Si vous voulez ne voir que les processus qui vous appartiennent par exemple, vous pouvez rediriger la sortie de `ps -ef` vers la commande `grep votre-login`.

Exercice 64 (Compter le nombre de fichiers dans un répertoire) Faites coopérer `ls` et `wc` pour compter le nombre d'objets qui se trouvent sur votre répertoire courant, puis dans `/usr/bin/` (le répertoire contenant la plupart des applications).

7.10 Outils spécifiques à l'Ensimag

En plus des commandes Unix standard, quelques petits programmes bien pratiques sont disponibles sur les machines de l'Ensimag (pour beaucoup, elles ont été réalisées par des étudiants). Ces commandes sont listés sur EnsiWiki, « Commandes spécifiques à l'Ensimag »).

Plus généralement, la page « *Portail :Ensimag* » d'EnsiWiki rassemble les pages du Wiki spécifiques à l'école, et quelques liens vers d'autres sites web liés à l'école.



Jeu de piste

Ce chapitre vous a donné de bonnes bases en ligne de commande Unix. Vous pouvez maintenant presque terminer le jeu de piste. Si l'étape F1 vous pose problème, passez au chapitre suivant, il devrait vous débloquer.

Chapitre 8

Utilisation du *shell Bash*

Le *shell* (`bash`) est un interpréteur de commandes disposant d'un certain nombre de caractéristiques qui en font un outil confortable à utiliser. C'est le shell par défaut à l'Ensimag.

Les autres *shells* très utilisés sont `tcsh`, qui était un peu l'ancêtre des *shells* très « confortables » et `zsh` qui offre des fonctionnalités inédites par rapport aux autres *shells*.

Les possibilités plus avancées décrites ici pour `bash` sont également présentes dans les autres shells, mais pas nécessairement de la même façon.

Il est possible de changer de nombreux paramètres du shell en positionnant la valeur de certaines variables spécifiques (cf section 8.4.2).

8.1 Les 1001 manières d'entrer une commande

Votre shell propose plusieurs façons de rentrer une commande. Le but est de minimiser le temps qu'il faut pour cela. Les fonctionnalités suivantes sont donc des aides précieuses. Il vous est recommandé de lire et d'expérimenter attentivement cette partie.

8.1.1 La complétion des noms

Pendant la frappe d'une commande, le shell `bash` est capable de terminer de lui-même les noms de fichiers dès qu'il dispose d'un début non ambigu. Pour donner au shell l'ordre de terminer un nom de fichier, il suffit de taper le caractère `TAB` (avec , une touche tout à gauche, la troisième en partant du haut)

Supposons que le répertoire courant contient les fichiers suivants :

```
brouillon  chap1  courrier  hello.py
lien-vers-shell  LISEZ_MOI  livre  shell
```

Si on veut faire `emacs chap1`, il suffit de taper `emacs ch` , le shell complètera en `emacs chap1`.

Si au moment de l'émission de la commande `TAB`, le début du nom est ambigu, le shell affiche tous les noms possibles puis réaffiche le début de la commande. Il suffit de taper juste ce qu'il faut de caractères pour rendre le nouveau début non ambigu et de retaper  pour obtenir le nom désiré.

Exercice 65 (Complétion) Créez un fichier avec un nom à rallonge avec la commande `touch fichierAvecUnNomTresTresLong.txt`. Essayez de l'ouvrir avec `emacs`, en utilisant la complétion, en appuyant sur maximum 7 touches.



Le saviez-vous ?

Complétion avancée avec bash

En fait, la complétion que nous venons de voir ne s'applique qu'aux noms de fichiers, mais il est possible de paramétrer toutes sortes de complétions intelligentes. Cherchez `bash_completion` dans votre moteur de recherche préféré, ou lisez la section 8.6.3...

8.1.2 Gestion de l'historique des commandes

Il arrive très souvent que l'on ait à utiliser tout ou partie du texte d'une commande précédemment tapée, soit qu'on veuille refaire une commande, soit qu'on veuille légèrement modifier une commande erronée. Le *shell bash* permet de réaliser cela de manière très agréable. Toutes les commandes que l'on tape sont mémorisées dans un fichier d'historique (par défaut, `.bash_history`).

Les flèches de curseur  et  permettent de revenir en arrière ou d'avancer dans la liste.

En `bash`, une manière pratique de naviguer dans l'historique est d'utiliser  +  puis *texte-a-rechercher*, à la manière d'`Emacs`, qui recherche *texte-a-rechercher* dans l'historique. Il est possible de taper plusieurs fois sur  +  pour revenir aux occurrences précédentes.

Exercice 66 (Retrouver une commande) Essayez de retrouver, en utilisant  +  une commande contenant une chaîne particulière que vous avez tapée aujourd'hui dans le terminal.



Le saviez-vous ?

history

La commande `history` permet d'obtenir la liste des dernières commandes avec leur numéro. Il est possible de refaire une commande en utilisant les raccourcis `!numero-d-historique`, `!debutChaine` et `!?chaine?`.

8.1.3 La convention tilde

1. Tous les mots d'une commande sont scrutés pour déterminer s'ils commencent par le caractère `~`. Si oui, le mot entier (ou le début du mot jusqu'au premier caractère `/`) est interprété comme un nom d'utilisateur, et remplacé par le *répertoire d'origine* de cet utilisateur (cf section 4.3.6). Supposons qu'on désire lire le contenu du fichier `data` se trouvant dans le *répertoire d'origine* de l'utilisateur `rouverol`, on peut émettre la commande :

```
less ~/rouverol/data
```

que `bash` transformera en :

```
less /users/rouverol/data
```

2. Si le nom d'utilisateur suivant le signe `~` est vide, le nom pris par défaut sera celui de l'utilisateur lui-même. Supposons que l'utilisateur `cassagne` désire lire un fichier se trouvant dans son *répertoire d'origine*, il peut émettre la commande :

```
less ~/data
```

que le shell transformera en :

```
less /users/cassagne/data
```

8.2 Parler de plusieurs fichiers à la fois : les *wildcards*

Les wildcards sont des métacaractères qui peuvent être utilisés pour parler de plusieurs fichiers à la fois. Dans ce cadre :

- le caractère « `*` »

remplace une chaîne (potentiellement vide) de caractères quelconques. Par exemple : `co*de` remplace `code`, `corde`, `commande` ;

- le caractère « `?` »

remplace un et un seul caractère quelconque. Par exemple : `co?de` remplace `corde`, mais pas `code` ni `commande` ;

- l'expression « `[abc]` » remplace soit `a`, soit `b`, soit `c`. Par exemple : `fichier.od[pts]` remplace `fichier.odt`, `fichier.odp` ou `fichier.ods` ;

- l'expression « `[a-f]` » remplace un caractère entre `a` et `f`. Par exemple : `chap[1-3]` remplace `chap1`, `chap2` ou `chap3` ;

- l'expression « `[^abc]` » remplace n'importe quel caractère autre que `a`, `b` ou `c`. Par exemple, `to[^t]o.txt` remplace `tooo.txt`, `toxo.txt`, mais pas `toto.txt` ;

- il existe d'autres caractères spéciaux, plus d'informations dans le manuel de votre shell (cf. section 7.5) ;

Exercice 67 (Listage) *La commande suivante imprime sur l'écran le contenu d'un ou plusieurs fichiers :*

```
cat noms_de_fichiers
```

Utilisez cette commande et les wildcards pour imprimer le contenu de plusieurs fichiers à la fois. Si la commande affiche trop de texte, essayez par exemple avec `ls` à la place de `cat`.

Si on veut parler explicitement d'un fichier dont le nom comporte un caractère spécial (comme `*`, par exemple, si un fichier s'appelle vraiment `to*to.txt`, c'est rare, mais possible!), il suffit d'entourer son nom avec des guillemets simples ou doubles. Par exemple, `ls *.txt` affiche tous les fichiers dont le nom termine par `.txt`, alors que

```
ls '*.txt'
```

et

```
ls "*.txt"
```

affichent le fichier dont le nom est « *.txt ».

8.3 Lancer plusieurs commandes en même temps

Le système que nous utilisons permet bien entendu de lancer plusieurs applications en même temps. Depuis l'interface graphique, c'est très simple, on lance plusieurs applications depuis le menu.

Mais depuis un terminal, c'est un peu différent. Par exemple, lancez un terminal, et dans ce terminal, lancez la commande `emacs`. Vous pouvez utiliser `Emacs`, mais vous n'avez plus la main dans le terminal.

8.3.1 Première solution : Control-z, bg

Première solution : suspendre `Emacs`, ce que l'on peut faire avec `Ctrl` + `z` dans le terminal depuis lequel on a lancé `Emacs`. Pour qu'`Emacs` continue à s'exécuter normalement, il faut l'autoriser à tourner « en tâche de fond », avec la commande `bg` (comme `back ground`).

Exercice 68 (Tâches de fond) Dans un terminal, lancez la commande `emacs`. Revenez au terminal sans fermer la fenêtre d'`Emacs`, puis faites `Ctrl` + `z`. Lancez une autre commande dans le terminal, par exemple, `ls`. Essayez maintenant de vous servir de la fenêtre `Emacs` : plus rien n'a l'air de marcher, `Emacs` est suspendu. Revenez dans le terminal et faites `bg`. Vous gardez la main dans le terminal (i.e. vous pouvez exécuter d'autres commandes), mais `Emacs` reprend vie. Essayez maintenant la commande `fg` : on est revenu dans la situation initiale.

En pratique, si vous ne devez retenir qu'une chose : quand vous voulez reprendre la main dans un terminal, faites `Ctrl` + `z` puis `bg`.



Le saviez-vous ?

Tâches de fond et applications en mode texte

Les commandes `fg` et `bg` sont en fait très utiles quand on utilise des applications en mode texte (`vim`, `man`, `~ less~...`) et qu'on souhaite en lancer plusieurs dans le même terminal : on revient au shell avec `Ctrl` + `z`, on peut lister les applications qui s'exécutent dans le terminal courant avec `jobs`, et basculer vers cette application avec `fg` ou `fg %1`, `fg %2...`

8.3.2 Deuxième solution : lancer en tâche de fond avec « & »

Voici une autre solution pour lancer une application en tâche de fond : ajouter le caractère `&` à la fin de la ligne de commande. On se retrouve alors directement dans la même situation qu'en ayant fait `Ctrl` + `z` puis `bg`.

Exercice 69 (Lancement en tâche de fond) Lancez la commande

```
emacs &
```

puis dans le même terminal, lancez une autre commande comme `ls` : elle devrait s'exécuter normalement. Essayez d'utiliser la fenêtre **Emacs** que vous venez de lancer, puis fermez cette fenêtre.

8.4 Configuration de Bash : le fichier `.bashrc`

8.4.1 Un fichier lu à chaque lancement

La plupart des commandes permettant de configurer le *shell* `bash` s'appliquent au shell courant.

Exercice 70 (Alias) Par exemple, essayez de lancer la commande suivante dans un terminal.

```
alias toto=ls
```

À partir de maintenant, quand vous entrerez la commande `toto`, elle se comportera comme `ls`. Fermez maintenant ce terminal puis ouvrez-en un autre. L'effet de la commande a disparu.

Il serait pénible de devoir entrer à la main la configuration du shell à chaque lancement de terminal, c'est pourquoi `bash` prévoit un mécanisme d'initialisation : le fichier `.bashrc`, situé à la racine de votre compte, sera lu à chaque démarrage de shell (en particulier, à chaque fois que vous ouvrirez un terminal).

Tout ce que vous mettrez dans ce fichier pourrait être entré à la main directement dans un terminal. Vous pouvez vous amuser à y mettre des commandes interactives ou bien des commandes affichant du texte, mais ne gardez pas ce genre de chose dans votre configuration : ceci pourrait gêner le fonctionnement de certaines applications.

Les deux grands types de configuration sont l'affectation de valeurs à des variables du shell et la création d'alias de commandes.

8.4.2 Les variables

Le shell gère des variables, c'est-à-dire des couples (nom,valeur). Une variable du shell n'a pas besoin d'être déclarée, elle prend existence dès qu'on lui affecte une valeur. Pour affecter une valeur à une variable, il faut émettre la commande :

```
nom-de-variable=valeur
```



Attention !

Les blancs sont significatifs.

Ne pas mettre de blanc entre *nom-de-variable* et le signe =, ni entre le signe = et *valeur*.

Exemple :

```
dir1=/usr/include
```

La référence à la valeur d'une variable se fait par `$nom-de-variable`. Exemple :

```
echo $dir1
/usr/include
```

Si on désire connaître l'ensemble des variables et leurs valeurs à un instant donné, on utilise la commande `set`. Exécutez cette commande, vous voyez que le système a créé un certain nombre de variables en plus de celles que vous avez pu créer.

Les variables sont un moyen agréable de réaliser des abréviations. Supposons que l'on prévoie d'avoir à plusieurs reprises à référencer le fichier `/usr/include/stdio.h`, on peut se définir la variable `std=$dir1/stdio.h` (en supposant qu'on a déjà affecté la variable `dir1` comme indiqué plus haut). Il sera ensuite agréable d'utiliser `$std` à la place du nom véritable.



Le saviez-vous ?

Les variables, par défaut ne sont pas **exportées** vers les commandes lancées par le *shell*. Pour qu'elles soient exportées, c'est-à-dire visibles par les autres applications, il faut ajouter le mot clef `export` :

```
export dir1=/usr/include
```

on dit alors que la variable est une variable d'environnement.

8.4.3 Alias

On peut définir un nom comme étant un alias (c'est-à-dire une abréviation) pour une chaîne de caractères. Si par exemple, on a souvent besoin d'exécuter un programme de nom `analyse` on peut émettre la requête suivante :

```
alias ana=analyse
```

ce qui a pour effet de faire mémoriser par `bash` que lorsque le **premier** mot d'une commande est `ana`, il doit le remplacer par `analyse`. Une exécution du programme `analyse` pourra donc se faire par :

```
ana fic1 fic2
```

(en supposant que `fic1` et `fic2` soient deux arguments nécessaires à l'exécution du programme.)

On peut faire en sorte qu'un alias soit remplacé par une chaîne contenant des blancs, mais cela nécessite de mettre entre *quotes* le texte de l'alias. Supposons que l'on prévoie d'avoir à recompiler plusieurs fois le calcul de la factorielle de 10 en Python avec `fact.py`, on peut se définir un alias de la manière suivante :

```
alias fact10='python3 fact.py 10'
```

on peut ensuite lancer l'exécution en tapant simplement :

```
fact10
```

Lorsque l'expansion d'un alias a été réalisée, le premier mot de la commande obtenue n'est plus susceptible de subir à nouveau une expansion d'alias. On peut donc sans crainte définir des alias « récursifs », comme, par exemple :

```
alias ls='ls -F'
```



Le saviez-vous ?

alias

La liste des alias déjà définis est obtenue avec la commande `alias`.

8.5 Où sont rangées les commandes ?

Nous avons vu qu'un programme était un fichier exécutable, et nous en avons même créé un petit en utilisant le langage Python. Nous avons aussi vu qu'entrer des commandes dans un *shell* exécutait des programmes. Il nous reste à comprendre comment le *shell* trouve les programmes à exécuter quand on entre des commandes.

Une première solution, que nous avons utilisée quand nous avons exécuté notre programme Python était d'entrer le chemin de l'exécutable, relatif ou absolu. Par exemple, pour exécuter un programme qui se trouve dans le répertoire courant, on fait

```
./programme
```

et pour exécuter Firefox, on pourrait entrer `/usr/bin/firefox`. Mais nous avons vu qu'entrer seulement `firefox` revenait au même.

Pour le savoir, le shell va regarder le contenu de la variable `$PATH` : elle définit une liste de répertoires par défaut où le système va chercher les exécutables.

Exercice 71 (\$PATH) *Dans un terminal, lancez la commande*

```
echo "$PATH"
```

elle affiche le contenu de la variable \$PATH. Elle devrait commencer par quelque chose comme /usr/local/bin:/usr/bin:/bin.

Quand on entre une commande comme `firefox`, le *shell* va tester la présence d'un exécutable `firefox` dans chacun des répertoires. Comme `/usr/local/bin/firefox` n'existe pas (on peut le vérifier avec `ls`), le *shell* passe au suivant, `/usr/bin/firefox`, qui existe, donc le *shell* exécute ce fichier.

Pour savoir où se trouve une commande, on peut utiliser la commande `type`, comme ceci :

```
type nom-de-commande
```



Le saviez-vous ?

Commande interne

Certaines commandes, comme, par exemple `cd` sont en fait des **commandes internes** du shell. Une commande interne n'est pas un programme normal comme `firefox`, ou `emacs`, car c'est l'état interne du *shell* lui-même qui est changé.



Le saviez-vous ?

Programme compilé et programme interprété

Les programmes comme `ls`, sont compilés dans le jeu d'instruction du processeur. La machine peut donc les exécuter tel quel. Pour les programmes écrits en Python, ou en shell, ils sont interprétés, c'est-à-dire traduits à la volée en langage machine.

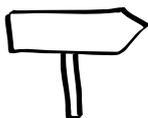
Le début du programme à interpréter, la première ligne, indique au système quel interpréteur à utiliser. Il commence par `#` puis `!` puis le *chemin-vers-l-interpréteur*. L'interpréteur, lui, est écrit en langage machine.

```
#!/bin/bash
```

ou bien

```
#!/usr/bin/python3
```

Exercice 72 (Où se trouvent les commandes?) *Où se trouvent les fichiers exécutables correspondant aux commandes `firefox`, `ls`, `cd`, `rm` ?*



Jeu de piste

Si vous n'aviez pas réussi à résoudre l'étape F1, vous devriez maintenant pouvoir le faire.

8.6 Exemples de configuration de bash

Voici quelques exemples de configurations classiques et/ou bien utiles de bash. En général, votre distribution ou votre administrateur système a déjà créé un fichier `.bashrc` pour vous : c'est une bonne idée de le lire, pour voir quelles fonctionnalités sont activées ou non, et adapter le tout à vos préférences.

8.6.1 Des alias pour le confort

La première utilisation des alias est de créer des raccourcis pour des commandes utilisées souvent. Parmi les classiques, on trouve par exemple (dans le `~/.bashrc`) :

```
alias ll='ls -alF'
alias la='ls -A'
```

Ainsi, entrer `ll` exécutera la commande `ls -alF` (i.e. affichera tous les fichiers y compris les fichiers cachés, avec les détails de permissions, appartenance et taille).

On peut aussi utiliser les alias pour redéfinir des commandes existantes, et donc modifier leur comportement par défaut. Par exemple (toujours dans le `.bashrc`) :

```
alias ls='ls --color=auto'
```

En utilisant la commande `ls`, on exécutera en réalité `ls --color=auto`, donc on verra une sortie en couleurs. En d'autres termes, cet alias est une façon de dire « je veux que la commande `ls` affiche des couleurs ».

8.6.2 Des alias pour éviter les erreurs

Trois alias fréquemment utilisés (vous pouvez les ajouter dans votre `.bashrc` s'ils n'y sont pas et si vous voulez en bénéficier) sont :

```
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
```

Ces alias demandent aux commandes `cp`, `mv` et `rm` de toujours utiliser l'option `-i`, qui demande une confirmation interactive avant d'effacer un fichier. Pour la commande `rm`, la demande de confirmation se fait à chaque fichier. Pour les commandes `cp` et `mv`, on aura une demande de confirmation avant d'écraser un fichier si la source existe déjà (par défaut, `cp fichier-source fichier-destination` détruit silencieusement *fichier-destination* s'il existait).

Pour éviter les demandes de confirmation, on peut au choix utiliser `\rm` (le `\` demande au shell d'utiliser la commande, et pas l'alias) ou `rm -f` (qui sera expansée en `rm -i -f`, l'option `-f` annulant l'effet du `-i`).

8.6.3 La complétion programmable

On a déjà parlé de l'utilité de la touche tabulation (`⇐⇒`) dans un shell pour compléter un mot en cours d'écriture. En général, la complétion se base sur les noms de fichiers : par exemple, si un fichier `rapport.tex` est présent dans le répertoire courant, on s'attend à ce que `pdflatex rapp ⇐⇒` complète en `latex rapport.tex`. Mais on peut aussi avoir envie d'une complétion plus intelligente. Par exemple, si on a également des fichiers `rapport.log` et `rapport.pdf` dans le même répertoire, on peut avoir envie que `evince rapp ⇐⇒` complète avec le fichier PDF, et que `pdflatex rapp ⇐⇒` complète avec le fichier `.tex`. Tout ceci est possible pourvu qu'on le demande à `bash`, en chargeant le fichier `/etc/bash_completion`. On peut le faire comme ceci :

```
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi
```

Dans cette configuration, la complétion peut aussi s'appliquer aux options des commandes connues de `bash`, par exemple `find . -na ⇐⇒` complète en `find . -name`.

Chapitre 9

Clients légers, poste de travail virtualisé, et serveurs

L'Ensimag dispose d'un ensemble de serveurs sur lesquels s'exécutent des machines virtuelles. Du point de vue utilisateur, une machine virtuelle est similaire à un PC classique, auquel on accède à distance. Du point de vue des administrateurs, une machine virtuelle est un programme, qu'on peut lancer en plusieurs exemplaires sur un serveur physique.

Les machines virtuelles sont accessibles :

- Depuis les clients légers de l'école. Pour les utilisateurs, ces clients légers sont similaires à des PC, mais il y a une grosse différence : avec les clients légers, les programmes lancés par l'utilisateur tournent sur un serveur, dans une autre pièce de l'école. Le client léger s'occupe principalement de l'affichage.
- Depuis n'importe quelle machine, y compris à l'extérieur de l'école. Ceci vous permet d'utiliser l'environnement informatique de l'école sur vos machines personnelles sans avoir besoin d'installer tous les logiciels. Les explications pour le faire sont disponibles sur la page « Accès distant aux machines virtuelles de l'Ensimag ».

Pour les détails techniques, voir la documentation sur EnsiWiki, « Clients légers à l'Ensimag ».

Chapitre 10

Accéder à l'Ensimag depuis l'extérieur et les ordinateurs portables

10.1 Les ordinateurs portables personnels à l'Ensimag

La majorité des étudiants de l'Ensimag possède un ordinateur portable personnel (95% en 2011). L'utilisation des machines personnelles dans le cadre de l'école est fortement encouragée : même si l'école fournit des machines, il est important que vous preniez l'habitude de gérer votre propre machine, et que vous vous familiarisiez avec l'utilisation de machines portables. L'école fournit un certain nombre de services (support informatique par les étudiants et pour les étudiants, wifi, salles de travail. . .) pour vous aider à utiliser vos machines dans les meilleures conditions possibles.

Vous trouverez quelques indications et conseils pour acheter un ordinateur portable sur EnsiWiki, « Conseils pour acheter un ordinateur portable ».

10.2 Les mails : cf. *Webmail*

Nous l'avons déjà vu en section 3.5 (page ??) : le *webmail* de l'Ensimag est un moyen simple d'accéder à ses e-mails depuis n'importe où dans le monde ! On peut aussi utiliser un client mail comme Thunderbird avec le protocole IMAP pour plus de confort.

10.3 Transférer des fichiers : sftp et ses amis

En plus du compte sur les PCs de l'école, vous avez un compte sur des serveurs, en particulier sur `pcserveur.ensimag.fr`. Nous allons l'utiliser ici pour quelques exercices à faire depuis les PCs. Toutes ces manipulations peuvent être faites depuis l'autre bout du monde, à condition d'être connecté au VPN Ensimag (cf. section 10.5.2).



Le saviez-vous ?

Travail à distance sous Unix

Les réseaux actuels, et Internet, sont nés sous Unix. Du coup Unix est prévu pour pouvoir travailler à distance comme si l'on travaillait localement. Cela se retrouve dans la plupart des couches logicielles. X11 en est un exemple, mais en fait il y a peu ou pas d'applications qui nécessitent d'être localement devant la machine, y compris pour la gestion d'un affichage 3D, le son, ou l'accès à un périphérique de stockage distant.

Il est très facile sous Unix de transférer des fichiers entre deux machines où l'on possède un compte. Pour cela il existe un protocole dédié à ces opérations qui s'appelle FTP. Vous l'avez peut-être utilisé dans votre navigateur pour télécharger des fichiers. FTP est en fait bien plus riche. Il permet de manipuler les répertoires distants, changer les droits des fichiers, effacer des fichiers, téléverser un fichier sur un serveur distant (l'inverse du téléchargement, *upload* en anglais), de reprendre un téléchargement interrompu...

Mais FTP transmet toutes les informations en clair sur le réseau, y compris les mots de passe. C'est pour cela que la plupart du temps, c'est sa version chiffrée qui est utilisée à sa place : SFTP

Il existe de nombreux clients SFTP textuels (`sftp`, `lftp`, `ncftp`) ou graphiques (Konqueror, Nautilus) y compris pour d'autres systèmes d'exploitation (`sftp`, WinSCP, FileZilla).

Lorsqu'il s'agit uniquement transférer un fichier ou un répertoire, la commande `scp` permet de le faire simplement avec une syntaxe proche de `cp`.

Exercice 73 (Copie de fichier) *Créez localement un fichier `creeSurPC` et copiez-le sur `pcserveur` avec la commande :*

```
scp creeSurPC \
  votre-login@pcserveur.ensimag.fr:copieDistante
```



Le saviez-vous ?

`rsync` : un `scp` amélioré

L'avantage de la commande `scp` est qu'elle est disponible presque partout, mais sur les machines sur lesquelles elle est disponible, la commande `rsync` fait en fait tout ce que sait faire `scp`, et bien plus !

10.4 Accéder directement au serveur Unix : SSH

Unix permet de tout faire en ligne de commande dans un *shell*. Il est donc possible de tout faire à distance, pourvu que l'on ait accès à un *shell* sur la machine visée. C'est le rôle du protocole SSH, utilisable via la commande cliente `ssh`.

Exercice 74 (La connexion à pcserveur) *Dans un terminal, connectez vous à pcserveur par la commande `ssh votre-login@pcserveur.ensimag.fr` (ou simplement `ssh pcserveur.ensimag.fr` vu que votre nom de login est le même sur les deux machines).*

En pratique, le contenu de vos comptes (i.e. les fichiers et répertoires qui s'y trouvent) sont partagés entre les PC et pcserveur, donc vous verrez la même chose en local et en distant.

Si vous faites la même manipulation depuis une machine autre que les PCs de l'Ensimag (par exemple votre ordinateur portable personnel), vous pourrez vérifier par des commandes sur les fichiers `ls`, `cp`, `mkdir`, `mv` que vous n'êtes pas dans le même compte.

Lors de la première connexion, `ssh` vous indique qu'il ne connaît pas cette machine et vous demande si vous voulez en garder la signature. C'est normal pour la première fois, et il faut répondre `yes` !



Le saviez-vous ?

Les signatures

Les signatures des machines auxquelles vous vous êtes connectés sont stockées dans le fichier `.ssh/known_hosts`. Il est parfois nécessaire de supprimer une signature de ce fichier lorsque qu'elle a changé (ex. : réinstallation complète de la machine).

Il est aussi possible de lancer des commandes graphiques à distance et de les afficher localement, `ssh` s'occupant du transfert chiffré des commandes X11.

Exercice 75 (Connexion à pcserveur avec connexion graphique) *Dans le terminal précédent, déconnectez vous de pcserveur puis reconnectez vous en ajoutant l'option `-X` à la commande `ssh`.*

Lancez `emacs copieSurServeur` (ou votre éditeur préféré à la place d'`Emacs` !) pour éditer et modifier le fichier.



Le saviez-vous ?

Emacs, KDE, Gnome et ssh

En fait, Emacs gère directement SSH et la manipulation précédente pouvait être faite directement sur votre machine (Emacs s'occupant du transfert au moment de sauver). Ouvrez simplement le fichier `/votre-login@pcserveur.ensimag.fr:fichier.txt` (depuis Emacs, `C-x C-f`). De même les logiciels pour Gnome et KDE permettent l'utilisation d'URL du type `sftp://votre-login@pcserveur.ensimag.fr/` (à entrer à la place d'un nom de fichier dans le sélecteur de fichier).



Le saviez-vous ?

screen (ou tmux))

Il arrive parfois d'avoir besoin de lancer une commande mais sans attendre sa fin, de terminer votre session et revenir uniquement plus tard. C'est possible et facile avec la commande `screen`. Une fois la commande lancée, on détache `screen` en tapant `Ctrl` + `a` puis `d` et on se réattache avec la commande `screen -r`.

En utilisant `screen`, il est même possible de lancer une commande depuis une salle de l'Ensimag, et de reprendre la main dessus une fois rentré chez soi, depuis une autre machine !

Pour plus d'information, consulter le Wiki, « SSH ».

10.5 Accéder à Unix depuis Windows et vice-versa

Il existe de très nombreuses possibilités pour faire ces accès dans les deux sens. La plupart de ces méthodes sont relativement simples à mettre en œuvre, même si cela demande souvent d'installer des logiciels spécifiques.



Le saviez-vous ?

Le réseau

La configuration des réseaux utilisés est importante. Les méthodes suivantes ont des besoins légèrement différents. L'accès que vous voulez faire demande peut-être à reconfigurer des pare-feux, NAT, routeurs, ou bien il est même parfois simplement impossible.

10.5.1 Utiliser Windows à l'Ensimag

Les PC de l'école sont généralement en *dual boot* Linux ou Windows (et éventuellement d'autres systèmes), c'est-à-dire qu'on choisit au démarrage de la machine quel système on souhaite utiliser. Nous allons maintenant nous intéresser à l'utilisation de ces PCs avec le système Windows.

Nous ne présentons pas ici les applications installées sous Windows : vous connaissez sans doute déjà les plus utiles, et la quasi-totalité des TP que vous aurez à faire pendant votre scolarité se feront sur les serveurs Unix. Par contre, il est important que vous sachiez comment accéder à votre compte Unix depuis ces salles. Ces instructions vous serviront en salles machines, mais aussi pour accéder à l'école depuis chez vous pour ceux qui ont une machine personnelle qui tourne sous Windows.

Accéder aux serveurs avec SSH

Une première solution pour accéder à votre compte est d'utiliser SSH. Comme le veut la tradition sous Windows, il existe des interfaces graphiques conviviales pour accéder au serveur et à vos données.

Le client graphique installé sur les postes Windows de l'Ensimag est la version non-commerciale du client **SSH Secure Shell**, disponible par exemple ici : <https://www.sfsu.edu/ftp/win/ssh/>. D'autres outils très utilisés pour faire du SSH depuis Windows sont Putty et WinSCP, tous deux très faciles à installer.

Exercice 76 (ssh) Lancez *secure shell client* depuis le menu *démarrer*. Cliquez sur l'icône *quick connect*. Dans le champ *Host name*, entrez *pcserveur.ensimag.fr*, et entrez votre *login* dans le champ *User Name*. Validez.

L'outil va vous demander une validation puis votre mot de passe, et au final, vous afficher un interpréteur de commande.

Vérifiez avec quelques commandes (*ls*, *pwd*, *ps...*) que tout ce que vous entrez ici s'exécute bien sur le serveur Unix.

Transférer des fichiers avec SFTP

Jusqu'ici, vous avez pu exécuter des commandes sur *pcserveur*, mais pas récupérer des fichiers. Voici maintenant deux manières de transférer des fichiers avec le même client SSH.

Exercice 77 (Transférer des fichiers) Dans le menu *Window* de *SSH Secure Shell*, choisissez *New File Transfer*. Vous obtenez une fenêtre à deux colonnes : à gauche, vos fichiers locaux, à droite, les fichiers de *pcserveur.ensimag.fr*. Vous pouvez copier des fichiers en faisant glisser à la souris.

Deuxième solution : fermez toutes les fenêtres *SSH Secure Shell*, puis choisissez *Secure File Transfer Client* depuis le menu *démarrer*. Vous obtenez la même fenêtre que ci-dessus.

Monter directement son répertoire avec SMB

Les solutions présentées ci-dessus ont l'avantage d'être réalisables depuis n'importe quelle machine. Sur un réseau local, il y a une autre solution, sans doute un peu plus simple : monter le disque du serveur Unix comme un lecteur réseau sous Windows. On y accédera ensuite comme un répertoire local avec l'explorateur de fichiers.

Exercice 78 (Montage du compte Ensimag) L'invite de connexion au serveur de fichiers se lance automatiquement à la connexion sur le PC Windows. En cas de problème avec le montage proposé automatiquement, vous avez également une icône sur le bureau pour le relancer : « Montage volume de données personnelles ». Une fois l'opération réussie, ouvrez le poste de travail depuis le menu *démarrer* : il doit afficher une icône en plus correspondant à votre compte Unix. Ouvrez ce répertoire et vérifiez que les fichiers sont bien les bons.

10.5.2 Autres solutions pour accéder à ses fichiers, exécuter des commandes à distance

PC virtuel : L'Ensimag dispose d'une infrastructure de virtualisation de bureau : vous pouvez accéder à l'environnement Ensimag (applications, fichiers, ...) à distance depuis n'importe quelle machine connectée à internet (Windows, Linux ou Mac OS). Nous en avons déjà parlé dans le chapitre 9

page ??.

SSH : La méthode la plus simple pour accéder à Unix depuis Windows est d'utiliser `ssh` et ses dérivés comme `scp`. Nous avons vu **SSH Secure Shell**, mais il existe bien d'autres clients Windows comme **Putty** (très simple à installer), **WinSCP** et **FileZilla**. On peut les trouver en partant de la page de <http://www.openssh.org> (section *Alternatives*).

SSH+X11 : si vous êtes sur Unix (ou Mac OS X), ou si vous avez installé un serveur X sur votre machine Windows, il est possible d'avoir un accès complètement graphique direct, quasiment identique à celui de votre terminal X. Il suffit pour cela d'utiliser l'option `-X` de `ssh` en ligne de commande. Pour installer un serveur X (et bien plus !) sous Windows, une solution est d'installer **Cygwin**, qui fournit un environnement de type Unix à l'intérieur de Windows.

VNC : accès graphique de tout vers tout (Unix, Windows, Java...), mais cela demande de lancer d'abord un serveur spécifique sur la machine à laquelle on veut accéder, puis de se connecter au serveur avec un client (éventuellement en utilisant `ssh` pour être sécurisé).

Partage de fichier ou d'imprimante avec SMB : Le partage de fichier avec le protocole de Windows est géré sous Unix par **Samba**.

VPN : il est possible de construire un réseau privé virtuel de façon à ce que vos applications se comportent comme votre machine était physiquement connectée dans l'Ensimag. Vous trouverez les explications sur l'intranet.

10.5.3 Plus d'information

Pour d'autres informations, par exemple des bonnes solutions pour synchroniser des répertoires entre plusieurs machines, ou comment monter son répertoire personnel depuis Mac OS X, rendez-vous sur EnsiWiki, « Travailler à distance ».

10.5.4 Les différences de codage des caractères

En informatique, on désigne sous le terme générique de caractère l'ensemble des lettres, chiffres et signes divers. On parle donc du caractère 'A', du caractère 'b', du caractère '+', etc. Dans un ordinateur, les caractères sont représentés sous forme de nombres : en codage ASCII, les lettres de A à Z ont les codes de 65 à 90, les lettres a à z ont les codes de 97 à 122, les chiffres de 0 à 9 ont les codes de 48 à 57...

Malheureusement, l'utilisation du codage ASCII entre différents systèmes n'est pas uniforme. Le fait de passer à la ligne utilise le caractère 10, ou « *new line* » sous Unix (y compris Mac OS X), le caractère 13, ou « *carriage return* » suivi du caractère 10 sous Windows. Il peut donc arriver qu'un programme windows (comme **Notepad**) ait du mal à lire un fichier texte produit sous Unix. Heureusement, beaucoup de programmes sont

capables de comprendre les deux formats (au moins Wordpad sous Windows, la plupart des éditeurs de textes sous Unix), et il existe des programmes pour passer de l'un à l'autre : `dos2unix` et `unix2dos`.



Le saviez-vous ?

Les extensions de ASCII

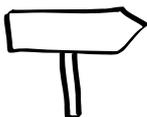
Les caractères ASCII ne définissent que 128 caractères. Même en codant tous les caractères uniquement avec 8 bits, on peut coder 256 valeurs.

Historiquement, Il existe donc un grand nombre d'extensions de ASCII pour tous les autres caractères utilisés par les différentes langues comme éâèùûâôçëï... en Français, incompatibles entre elles, et différentes d'un système à l'autre. Par exemple, pour les caractères du français on trouve la norme ISO-8859-1, également appelée latin1 sous Unix, CodePage 850 (originellement d'IBM) sous DOS, UTF-16 sous Windows.

Néanmoins, au moins sous Unix, la situation s'oriente vers l'utilisation du codage UTF-8, codage Unicode multi-octets compatible ASCII, capable d'encoder les caractères de presque toutes les langues vivantes écrites sur la planète. Le serveur `pcserveur` et les PC Linux que vous utilisez sont configurés pour utiliser UTF-8 par défaut.

Sous Unix, la commande `file` permet de deviner l'encodage d'un fichier, et les programmes `recode` et `iconv` permettent de passer d'un jeu de caractère à l'autre. Par exemple :

```
[me@centos ~]$ file toto.txt
toto.txt: UTF-8 Unicode text
[me@centos ~]$ iconv --from utf-8 \
    --to latin1 toto.txt -o titi.txt
[me@centos ~]$ file titi.txt
titi.txt: ISO-8859 text
```



Jeu de piste

Voilà, vous avez maintenant toutes les connaissances pour terminer la partie principale du jeu de piste (mais vous pouvez si vous le souhaitez vous attaquer à la deuxième partie du jeu de piste, plus orientée « Geek » et plus difficile, mais qui devrait vous apprendre encore beaucoup de choses!).

10.6 Et maintenant ?

10.6.1 Installer Linux sur un ordinateur personnel

Linux en tant que tel, n'est techniquement que le cœur du système. Autour de ce cœur se greffent de nombreuses applications comme X11 ou Firefox et dont beaucoup viennent du projet GNU (<https://www.gnu.org>) (Emacs, Vim, gcc, bash, ls...). On appelle cet ensemble une **distribution**. Leur grande force est que si vous avez un accès permanent à Internet, le catalogue de base des distributions grand-public est d'environ 10~000 à 20~000 applications de toutes natures, disponibles en quelques clics.

Il existe de nombreuses distributions différentes qui permettent une installation simple et facile sur un ordinateur personnel de nombreuses manières différentes : CD/DVD, clefs USB, réseaux, etc.

Il est vivement recommandé d'installer Linux sur sa machine personnelle : vous apprendrez beaucoup de choses, et vous gagnerez en confort en choisissant vous-même ce que vous installez et comment vous configurez votre machine. Pour vous aider dans cette tâche, un groupe d'étudiant est payé par l'Ensimag : les bug busters. Vous les trouverez sur EnsiWiki, « FAQ Bug Busters », ou en lisant les affiches en salles machines.

Si vous ne savez pas quelle distribution choisir, voici deux solutions :

- La distribution Ubuntu, très populaire auprès des débutants, et relativement simple à installer.
- La distribution CentOS, qui a l'avantage de vous donner le même environnement que celui de l'Ensimag. L'inconvénient de cette solution est que la distribution CentOS évolue moins vite qu'Ubuntu : vous aurez donc des logiciels moins récents, et surtout, les périphériques matériels récents ont peu de chance d'être supportés. En pratique, installer CentOS dans une machine virtuelle marche assez bien, mais l'installation sans machine virtuelle pose souvent problème.

L'environnement de l'Ensimag est en fait une distribution CentOS à laquelle ont été ajoutés un certain nombre de sources de paquets et de logiciels installés à la main. Pour plus de détails, consulter la page « Portail:Centos ».

En pratique, beaucoup d'étudiants choisissent d'avoir une distribution autre que celle de l'Ensimag sur leur machine personnelle. Dans ce cas, il reste important de tester vos TP et projets sur l'environnement Ensimag avant de les rendre (en utilisant les machines virtuelles ou les salles machines de l'Ensimag).

Linux cohabite avec les autres systèmes d'exploitations installés sur la machine. Son installation terminera en mettant en place un menu de démarrage qui vous proposera de choisir entre les différents systèmes installés.



Le saviez-vous ?

Les pilotes du matériel

La quasi-totalité des matériels fonctionneront immédiatement, mais comme pour une installation de Windows, un certain nombre de matériels exotiques demandent des pilotes particuliers à installer après l'installation de base.

**Attention !****Lire les messages !**

L'installation est très facile, mais il faut quand même lire attentivement les messages pour vérifier que l'installateur fera ce que vous souhaitez.

En particulier, il ne faut pas demander une installation sur la totalité du disque dur si l'on souhaite conserver une installation pré-existante de Windows avec les données qui sont dedans.

10.6.2 Continuer à apprendre ...

Si vous avez lu ce document en entier, et si vous êtes arrivés au bout du jeu de piste, vous avez l'essentiel pour travailler à l'Ensimag. Mais l'apprentissage ne s'arrête pas là : il reste beaucoup d'outils à apprendre, et beaucoup de pratique à acquérir.

Vous apprendrez certains de ces outils dans votre scolarité à l'Ensimag, mais il est indispensable d'être curieux, et de compléter l'apprentissage scolaire par une démarche personnelle. Nous vous proposons quelques pistes pour apprendre quelques outils très utiles, sur EnsiWiki « Ensimag niveau avancé ».

Index

!, 74
< (redirection), 69
> (redirection), 69
*, 75
-help, 64
.bashrc, 77
?, 75
@, 14
\$, 78
&, 76

a2ps, 63
accents, 14, 91
acroread, 44
alias, 77, 78
alpine, 28
arborescence, 33
ascii, 91
Atom, 47

bash, 73
bg, 76
bureau virtuel, 16

caché (fichier), 53
cal, 31
cancel, 62
cd, 35
chemin absolu, 36
chemin d'accès, 34
chemin relatif, 36
chmod, 55
clavier azerty, 14
commande, 16, 31
 interpréteur, 32
 interpréteur de, 16
complétion, 73
copier-coller, 15
cp, 40
Ctrl-c, 68

df, 56

distribution, 92
dos2unix, 90
du, 56

emacs, 48
ensiwiki, 7
entrées-sorties, 68
eog, 44
evince, 44
excel, 43

fg, 76
fichier, 34
 caché, 53
file, 91
find, 58
firefox, 23
ftp, 86

gimp, 44
grep, 60
gthumb, 44
gzip, 61

help, 64
historique, 57, 74
history, 74
home, 37, 74

iconv, 91
imap, 27
impression, 62
info, 64

jobs, 76
jockey, *voir* wildcard

kill, 67

latex, 44
less, 56
LibreOffice, 43
lien symbolique, 52, 58

Linux, 13, 92
ln, 58
lp, 63
lpr, 63
lpstat, 62
ls, 34, 39, 51
lsof, 65

man, 64
mkdir, 39
mot de passe, 19
mutt, 28
mv, 41

nautilus, 34

Office, 43
oocalc, 43
oodraw, 43
ooimpress, 43
oowriter, 43
OpenDocument, 43
OpenOffice.org, 43

passwd, 20
PATH, 79
PDF, 25, 44
pdf2ps, 63
pkill, 68
PostScript, 63
Powerpoint, 43
processus, 65
programme, 65
prompt, 16, 32
ps, 65
ps2pdf, 63
pwd, 53

répertoire, 34
 courant, 53
 parent, 53
 racine, 38, 54
recode, 91
redirections, 68
rm, 40
root, 38
rsync, 86

scp, 86
screen, 87

sftp., 86
shell, 16, 73
ssh, 86
ssh, 88

tar, 61
tcsh, 73
terminal, 16
thunderbird, 26
tilde, 74
top, 65
type, 79

unicode, 91
Unix, 13
unix2dos, 90
users, 65
UTF-8, 91

variable, 77
 d'environnement, 78
vim, 48

wc, 69
webmail, 26
wiki, 7
wildcard, 75
Windows, 88
Windows XP/Vista, 91
Word, 43

xkill, 68
xterm, 17

zip, 62
zsh, 73