Introduction a UN*X et à la shell bash

Bruno Böttcher

bboett @ adlp.org
http://www.nohkumado.eu

25 octobre 2020

Résumé

Il s'agit d'un brouillon, je vous serais très reconnaissant de m'envoyer des critiques **constructives** , avec si possible la partie du texte corrigée inclue, pour faire cela télé-chargez ce fichier en mode HTML et appliquez les corrections (wget http ://www.nohkumado.eu/docs/introBash/index.html).

Remerciements :

— Bruno Krimm, pour avoir indirectement induit l'intro de base à UN*X
— Raymond Ostertag (raymond.ostertag@fairesuivre.com) pour avoir corrigé mes (nombreuses) fautes d'orthographe et apporté des suggestions pertinentes.
— mon fils Nathan, pour relecture

Table des matières

Chapitre 1

Un*x pour les nuls !

Pour les personnes qui n'ont jamais approché un ordinateur, et qui débutent sous Un*x, le début est, dans la plupart des cas, particulièrement pénible. Donc nous commencerons une section au plus bas niveau possible : celui de l'ingénu.

1.1 Bienvenue sous UN*X.

Chose curieuse, allez vous dire, pourquoi est-ce que Un*x est écrit avec une étoile au milieu du nom, alors que tout le monde sait, qu'il devrait s'y trouver un 'i'. C'est pour signaler que 'UNIX' est un nom protégé, et correspond à un produit commercial bien défini, c'est pour cela que tous les autres systèmes d'exploitation affiliés ont des noms différents (BSD, Solaris, SCO-Unix, Linux etc.).

Donc, qu'est-ce que UN*X ? Il s'agit d'un des nombreux systèmes d'exploitation qui permettent à des utilisateurs d'utiliser un ordinateur, une machine qui ne comprend que des séquences de chiffres, de manière facile et productive. Un système d'exploitation peut être décrit comme un oignon, couche sur couche, on passe du coeur, que des spécialistes chevronnés peuvent utiliser directement, à l'interface graphique, la partie que l'utilisateur verra en premier.

Dans les sections qui suivent, nous allons décrire rapidement, ce qui constitue un ordinateur, et nous approcher par les couches basse pas à pas à l'interface utilisateur. Cette compréhension facilitera l'utilisation des ordinateurs en général.

1.2 Premiers concepts : qu'est-ce qu'est un ordinateur ?

À en croire la publicité un ordinateur n'est composé que d'un processeur, ce terme est devenu de nos jours très courant, et la plupart savent qu'il s'agit là du cœur du système, la performance de ce dernier ayant une certaine influence sur le système complet. Ce que beaucoup de gens semblent ignorer, c'est qu'il existe tout un foisonnement de marques et de types de processeurs différents.

1.2.1 Qu'est-ce qu'un processeur ?

Il s'agit d'une puce électronique sur laquelle différents composants sont mis ensemble, les parties les plus importantes sont :

- l'ALU la partie qui fait des opérations d'arithmétiques booléennes (opérations sur des chiffres à base de 2),
- les registres, qui stockent les données et résultats entre deux opérations soit après les avoir lues de la mémoire soit avant de les y réécrire,
- les décodeurs d'instructions qui transforment les commandes envoyées au processeur en séquences traitables par l'ALU,
- le cache, mémoire tampon, qui en règle générale fonctionne à une vitesse très proche de celle du processeur, mémoire qui se branche entre la mémoire centrale et le processeur pour accélérer le rythme des opérations en stockant une partie des transferts entre mémoire et processeur, si le processeur demande des données dans la mémoire qui se trouvent encore dans le tampon, le tampon répondra à la place de la mémoire,
- des processeurs dédiés (appelés coprocesseurs) implémentant certaines fonctions spécifiques, le plus connu étant le coprocesseur pour calcul à virgule flottante, ou un coprocesseur graphique.

1.2.2 Mémoire vive

Mais, un ordinateur est composé encore de beaucoup d'autres éléments. Ainsi un autre élément important est la mémoire vive, c'est là que seront stockés programmes et données dont aura besoin le processeur dans l'immédiat. La quantité de mémoire présente, la vitesse avec laquelle le processeur pourra récupérer ses informations (produit entre la largeur du mot récupéré et la vitesse maximale avec laquelle on peut récupérer ce mot), détermineront l'efficacité du système, donc sa performance. Techniquement parlant, dans la plupart des cas il s'agit de matrices de transistors qui stockent l'information dans des capacités.

1.2.3 Mémoire de masse

Autre élément très sous-estimé : le système de stockage de masse, autrement dit, les disques durs, la mémoire étendue, qui stockera de manière durable (espérons-le) des données et des programmes, qui seront chargés ou déchargés à la demande du processeur dans la mémoire vive. Le système disque utilisé influera de façon majeure sur les performances du système.

On distingue le ou plutôt les disques qui sont des disques magnétisables, comme les cassettes audio, et le contrôleur qui s'occupera de faire bouger la tête de lecture et de lire ou écrire des données.

Ainsi pour la plupart des acheteurs n'existent que les disques dits SATA évolution d'IDE. L'IDE était la variante la plus bon marché pour piloter des disques, d'intelligence très limitée, ce système laisse au processeur central le soin des détails

du traitement des données. Ceci est défendable pour des systèmes qui n'ont que des besoins réduits, ou qui tournent sous des systèmes d'exploitation mono-tâche ou pseudo-multi-tâche (comme p.ex. MS-Windows 3.x ou 9x).

Le SATA a essayé de rajouter un peu d'intelligence sans vraiment réussir.

Ceux qui voudront tirer toute la puissance possible d'un ordinateur, utiliseront des sous-systèmes disques intelligents et des systèmes d'exploitation multi-tâche. Un système disque de ce type est le système SCSI. Ce système s'occupe tout seul des transferts disque mémoire, ou disque disque, laissant au processeur le loisir de s'occuper d'autre chose.

Parallèlement, vu que ce système est surtout utilisé par des professionnels, les disques durs eux-mêmes ne sont pas les mêmes que pour les systèmes de type IDE, ainsi ces disques tournent dans la plupart des cas de deux à trois fois plus vite que leur contreparties IDE, sont beacoup plus lourds, pour mieux encaisser vibrations et chaleur, ce qui les rend plus chers (et aussi plus bruyants).

1.2.4 Système d'affichage

Pour que l'ordinateur soit utile, souvent, on a besoin qu'il affiche de manière textuelle ou graphique des résultats. Pour cela existe le sous-système graphique, représenté par la carte graphique insérée dans l'ordinateur et l'écran, piloté par cette dernière.

Alors qu'au début des ordinateurs des sorties purement textuelles étaient suffisantes, sur des écran noirs et blancs avec 80 caractères en largeur et 25 à 40 lignes de hauteur, de nos jours on demande de plus en plus de graphisme.

Or le fait d'afficher des données de manière graphique, de mettre en place une interface utilisateur graphique, nécessite des ressources importantes. C'est pour cela que les cartes graphiques deviennent de plus en plus puissantes, étant en fait eux mêmes des ordinateurs complets, spécialisées dans le traitement de données graphiques. Ceci afin de décharger au maximum le processeur des tâches de l'affichage, libérant ainsi des ressources pour d'autres tâches.

1.2.5 Système d'entrée/sortie

Des données il faut les entrer d'une manière ou d'une autre, les outils de saisie les plus connus sont évidemment le clavier et la souris ("mulot"), mais il ne faut pas oublier les liaisons série (à travers lesquelles p.ex. les souris transmettent leurs informations à l'ordinateur), les liaisons parallèles (plus rapides, utilisées par les imprimantes notamment), des liaisons réseau (à travers des cartes réseau) et plus récemment des ports USB, système qui essaye d'unifier les différents périphériques au travers d'un système de connectivité unique et standardisé.

1.2.6 Mémoire morte

C'est de la mémoire à priori in-modifiable, appelé souvent BIOS, qui renferme une séquence de code permettant de faire un test, pour le souvent sommaire, des composants, d'initialiser les sous-systèmes (comme les cartes d'extensions) et de lancer le ou les systèmes opératifs à l'allumage ou redémarrage de l'ordinateur.

1.2.7 Le bus système

Là aussi une partie souvent sous-estimée du système, c'est le liant du tout, plus ce bus est intelligent, plus grande est la largeur de celui-ci, plus haute est la fréquence de fonctionnement, plus rapide sera l'ordinateur.

Ainsi p.ex. les bus de type PCI ont permis de s'affranchir de certaines des contraintes connues avec les bus de type ISA qui les ont précédés dans les architecture dites PC (p.ex. les interruptions fixes, rajout de cartes "à chaud", etc.).

Là aussi on verra aisément la différence entre les systèmes dits de grand public et les systèmes professionnels, ainsi p.ex. récemment les systèmes de type PC ont implémenté des bus véhiculant des mots d'une largeur de 64bits à une fréquence maximale de 130MHz, pendant que des systèmes de type DEC-Alpha utilisent depuis quelques années des bus de 200MHz et d'une largeur de 128bit.

1.2.8 Premier contact : le login

Après vous être assis devant l'ordinateur ou le terminal (le truc avec un écran et un clavier), vous vous retrouvez selon le cas soit devant :

une console, écran typiquement noir, mode texte 80 colonnes par 25 lignes, avec une invite du genre :

```
Debian GNU/Linux 2.0 yoda tty1
yoda login:
```

un login X, écran typiquement gris ou affublé d'images ou couleurs variées, mais avec immanquablement au milieu normalement une boite avec l'invite 'login :' et 'password :'.

Après avoir tapé à l'invite votre identificatif, puis après la seconde invite tapé votre mot de passe pour accéder à votre compte le premier contact avec ce système opératif sera à travers un 'shell' (coquillage en anglais, la ligne de commande). Que ce soit par la console, ou en mode graphique à travers de soi-disant xterms, le fonctionnement est strictement pareil. Ce shell permet de donner des commandes au système et normalement de visualiser le résultat de ces commandes. Ici en l'occurrence le shell s'appelle 'bash', (d'autres shell : zsh, ksh, csh, tcsh).

Ce shell à certaines caractéristiques visuelles, notamment il est coupé en deux, d'une part le début, une chaîne de caractères qui est imprimée par le système que l'on va appeler le 'prompt' et ensuite (souvent caractérisé par un gros carré noir, ou une ligne qui clignote) la partie ou l'utilisateur tapera ses commandes.

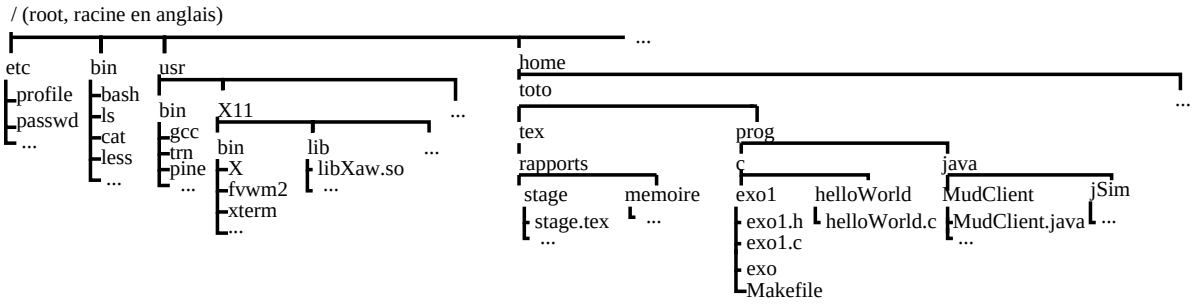

FIGURE 1.1 – Extrait d'un système de fichiers Un*x/Linux

Dès que vous aurez des problèmes, il est conseillé de consulter les 'man-pages'. Il s'agit d'un système de documentation. En tapant 'man man' vous aurez une introduction et les informations concernant son utilisation, de même que vous obtiendrez des informations sur bash en tapant 'man bash'. Il existe un deuxième circuit d'aides, apparu plus tard qui est appelé 'info', la aussi existe un tutoriel qu'il est impératif de suivre.

A chaque appel de commandes les arguments entourés de '[' et ']' sont optionnels et peuvent être omis.

1.3 Le système de fichiers

Il est important de comprendre, comment les données (fichiers, programmes, images etc.) sont organisées, et comment est-ce qu'on y accède.

En fait, il faut s'imaginer le système de fichiers comme un grand arbre avec autant de noeuds et de de feuilles que l'on veut (ce n'est pas tout a fait vrai comme ça, mais pour le début cela suffira).

Typiquement un système de fichiers sous Linux à la structure ébauchée dans l'image 1.1

Le premier noeud de cet arbre est typiquement appelé racine en français ou root en anglais, plus spécifiquement la racine s'écrit '/', en même temps ce symbole sera le séparateur de chemin, c'est à dire, qu'il séparera les noms des différents répertoires (noeuds de l'arbre) quand on composera des chemins d'accès.

À savoir que tout système Un*x, étant un système multi-utilisateur où chaque utilisateur possède son propre environnement, chaque utilisateur aura sa propre racine (par ex. l'utilisateur toto de l'image, aura comme racine /home/toto, attention, il y a des normes concernant ces nommages, mais p.ex. Apple ne les respecte pas, et les racines des utilisateurs y sont sous /Users).

Pour la nomenclature : il existe sur tout système Un*x un super-utilisateur, celui, qui fait l'administration et pour lequel n'existe aucun fichier caché. Cet utilisateur possède souvent le nom de 'root' lui aussi, et sur nombre de systèmes Un*x, sa racine est '/' justement (ce n'est pas le cas dans Linux).

Quand vous adressez un fichier dans cet arbre vous suivez en fait un tracé en

partant d'une des racines ou de votre localisation actuelle. Par exemple toto aimerait visualiser le fichier exo1.c qui se trouve quelque-part dans sa hiérarchie de fichiers, en partant de l'assomption qu'il a changé son répertoire courant ('pwd' en anglais, est en même temps une commande pour visualiser le répertoire courant) pour exo1, il a trois possibilités :

— less exo1.c (adressage relatif à la position actuelle)
— less ~/prog/c/exo1/exo1.c (adressage relatif à la racine de l'utilisateur)
— less /home/toto/prog/c/exo1/exo1.c (adressage absolu)

1.3.1 La partie physique du système de fichiers

il faut noter, que ce qui vient d'etre décrit jusque là, est la représentation logique du système de fichiers, qui nous en cache la réalité physique. En fait le système de fichiers est composé de partitions de disques durs (des bouts de ces disques), de disques durs entiers, de parties de la mémoire de l'ordinateur, de mémoires externes, comme disques durs externes, clefs USB, CD de données etc...

Chapitre 2

Utilisation de base d'un shell

2.1 Opérations de base sur le système de fichiers

2.1.1 Alias

Vous vous rendrez vite compte que vous utiliserez continuellement un certain nombre de commandes, souvent longues. Ces commandes peuvent être appelées par un 'alias', une abréviation. Pour cela vous définissez dans votre fichier '.bashrc' une ligne 'alias...'. Vous trouverez la syntaxe exacte dans le fichier '/etc/profile'. (Par exemple toto veut rapidement se rendre dans son répertoire de développement et définira un alias du genre `alias exo='cd prog/c/exo1'`).

Avant toute autre chose, quand vous arrivez sur un système inconnu, tapez alias sans argument, pour vous rendre compte, quels alias sont déjà définis par le système. Si un alias venait à vous gêner (style, vous en avez marre de taper une confirmation chaque fois que vous voulez effacer un fichier) vous pouvez accéder à la commande d'origine (si l'alias à le même nom qu'une commande Un*x standard, voir rm sur yoda, sinon le problème ne se pose pas) en préfixant l'alias par '\'.

2.1.2 Visualisation du contenu d'un répertoire

Pour visualiser le contenu de répertoires vous avez les commandes suivantes :

ls [path] , donne le contenu du répertoire indiqué par 'path'. Par défaut 'path' égale à '.' qui représente le répertoire courant. À la fin de chaque nom qui apparaît, peuvent se trouver des symboles. Le '/' dans tmp/ signifie qu'il s'agit d'un répertoire, '*' dans telnet* qu'il s'agit d'un exécutable et le '@' dans zcat@ qu'il s'agit non d'un fichier mais d'un lien sur un fichier et '~' qu'il s'agit d'un fichier de sauvegarde. L'affichage est trié selon l'ordre alphabétique, à savoir que les fichiers préfixes avec un '.' arrivent en tête (les fichiers cachés), suivent les fichiers commençant par une majuscule, puis les minuscules.

dir , (commande non UN*X) 2.1.1 alias de 'ls -lF', en outre du contenu du répertoire, cette commande donne les droits d'accès des fichiers. cela ressemble à :

```
drwxr-xr-x  34 root   users        3072 Feb  9 12:05 tmp/
```

la première colonne ('d'), donne l'information qu'il s'agit d'un répertoire, les trois suivantes ('rwx') donnent les droits d'accès de l'utilisateur (root dans ce cas) ici l'utilisateur a les droits de lire 'r', d'écrire 'w', et d'exécuter 'x'.

Les trois colonnes suivantes ('r-x'), donnent les droits des utilisateurs affiliés à un groupe (groupe users, dans ce cas).

Les trois dernières colonnes ('r-x'), donnent enfin les droits d'accès de tous ceux qui ne tombent pas dans les premières deux catégories.

Le chiffre qui suit donne le nombre de liens qui pointent sur ce fichier.

Les deux noms donnent respectivement le propriétaire du fichier et le groupe d'utilisateurs ayant éventuellement accès.

Suivent la taille du fichier, la date de dernière modification et le nom du fichier.

Il est à remarquer qu'un répertoire doit être déclaré exécutable.

Pour changer les droits d'accès aux fichiers voir 'man chmod' et 'man chgrp'

la , alias de 'ls -aF', en outre des fonctionnalités de 'ls' les fichiers commençant par un '.' sont affichés. Ces fichiers contiennent normalement des données personnelles pour les applications de l'utilisateur et sont habituellement cachés.

ll , est un alias 'ls -laF', donne en plus des fonctionnalités de 'la' les droits d'accès comme 'dir'.

2.1.3 Comment changer de répertoire

Après vous être authentifié (procédure de login, voir 1.2.8), vous vous trouverez automatiquement dans le répertoire racine de votre compte. Votre répertoire racine s'appelle aussi ~votre_nom_de_login. Pour changer de répertoire vous avez la commande 'cd '. Pour remonter d'un répertoire vous utiliserez le lien '..' ('cd ..'). Pour revenir à n'importe quel moment vers votre répertoire racine, tapez 'cd' sans argument.

2.1.4 Comment créer un répertoire

On crée des répertoires avec la commande 'mkdir nom-du-repertoire' (make directory en anglais).

2.1.5 Comment enlever un répertoire ou fichier

La commande est 'rm'. Par défaut 'rm' n'enlève que des fichiers, avec l'option '-r' la commande devient récursive et enlève donc aussi les répertoires. Utilisez avec précaution : une fois un fichier ou répertoire détruit il n'y a aucun moyen d'en récupérer les données !

2.1.6 Comment copier un fichier

Pour copier un fichier vers un autre lieu ou pour créer une copie avec un autre nom, on utilisera 'cp source destination' (ex. cp exo1.c exo1.h crée une copie de exo1.c nommée exo1.h dans le même répertoire, cp exo1/Makefile helloWorld/ copie le fichier Makefile se trouvant dans le répertoire exo1 dans le répertoire helloWorld).

2.1.7 Visualisation d'un fichier

Pour visualiser un fichier il existe plusieurs méthodes, la syntaxe est toujours la même, 'nom-de-commande-du-fichier-à-visualiser' :

cat envoie le fichier entier sur la console. Commande assez primitive et surtout utile en tant qu'entrée pour des scripts, ou des commandes imbriquées.

more , à la différence de cat, more présente le fichier en portions d'un écran, l'utilisateur tape <ESPACE> pour avoir la page suivante. Les commandes de recherche (commande '/expression, voir 2.2) permettent de rechercher les locations qui intéressent l'utilisateur. Pour quitter l'affichage en cours de route taper 'q' comme quit.

less , (boutade envers 'more'...) est plus évolué, en plus des fonctionnalités de 'more', il est possible avec les flèches d'évoluer librement dans le fichier

view , encore plus évolué que less, permet les manipulations avec la logique de vi, mais sans "lock" (blocage) du fichier, et donc sans la possibilité de changer le contenu

2.2 Édition/création d'un fichier (avec vim)

L'édition d'un fichier se fait avec 'vi' (d'autres éditeurs existent, mais celui-ci est pratiquement universel, si vi n'est pas un lien vers 'vim' protestez auprès de votre administrateur système). Il est connu que la relation entre convivialité et puissance d'un outil est inversement proportionnelle, nous pouvons affirmer que 'vi' est très puissant... Il est expressément recommandé de lire la documentation sur site et l'aide en ligne ' :help' avant d'essayer de s'aventurer dans l'édition de fichiers.

Pour couper court, il faut comprendre que vi à 3 modes de fonctionnement : le mode commande, le mode compatibilité 'ed' et le mode insertion. Or ce n'est que ce dernier mode qui permet la capture de texte. Demandez à un utilisateur UN*X de vous donner son .exrc (ou plustot .vimrc) qui contient des initialisations pour vi (notamment des réglages du genre 'set showmode', 'set showmatch' etc.).

2.2.1 Le mode insertion

Pour entrer en mode insertion vous tapez

i insère à l'endroit du curseur,

I insère en début de ligne,

a insère après le curseur,

A insère en fin de ligne,

o insère une nouvelle ligne en dessous de l'actuelle et démarre en mode insertion,

O insère une ligne au dessus de l'actuelle et démarre le mode insertion

Pour en sortir vous tapez <ESC>, d'ailleurs, dès que vous vous sentez perdus, et avant de faire n'importe quoi tapez cette touche plusieurs fois avant de continuer, ceci afin d'être sûr d'être en mode commande.

2.2.2 Le mode compatibilité ed

Ensuite pour sauvegarder le fichier vous entrez en mode 'ed' en tapant ' :' en partant du mode commande, puis 'w' (comme write), pareil si vous voulez quitter c'est ' :q', si vous avez modifié un fichier et que vous essayez de le quitter, il faudra forcer la sortie, si vous ne voulez pas sauvegarder ' :q!'.

D'autre fonctionnalités utiles dès le début du mode ed sont la recherche par mot-clé (en mode commande tapez '/' puis le mot clé, 'n' pour la prochaine référence) et la recherche et substitution : par exemple `:%s/toto/titi/g` qui remplacera (:s) sur tout le fichier ('%', sans c'est juste sur la ligne actuelle, vous pouvez mettre aussi des rayons d'action <ligne départ,ligne-fin>), cherche toto ('/toto/') et remplace par titi ('/titi/' les deux '/' au milieu fondant en un), globalement, c'est à dire toutes les occurrences sur la ligne traitée (sinon uniquement la première occurrence sur une ligne est changée, 'g' = global).

2.2.3 Le mode commande

La méthode la plus plus facile pour sortir d'un fichier en le sauvegardant éventuellement, c'est de taper 'ZZ' (<SHIFT>ZZ si vous préférez). D'autres commandes intéressantes en mode commande sont par exemple 'Q rayon-d-action' pour reformater du texte, et les commandes de motion qui ont leur origine du temps ou les terminaux n'avaient pas encore les flèches

i pour monter,

j pour aller à gauche,

k pour descendre et

l pour aller à droite.

Ensuite les commandes de motion qui peuvent être combinées avec d'autres commandes, et qui préfixées d'un chiffre n, sont exécutés n-fois :

w avance d'un mot

b recule d'un mot

{ avance d'un paragraphe

} recule d'un paragraphe

Ex. '4w' avance de quatre mots. Suivent ensuite les commandes pour modifier/enlever du texte :

x détruit le caractère sous le curseur

d détruit (toujours suivi d'un rayon d'action) donc : 'd2w' détruis les deux mots qui suivent, 'd}' détruis le paragraphe jusqu'à sa fin avec la fonction 'dd' détruit une ligne.

c change (toujours suivi d'un rayon d'action), donc : 'cw' change le mot, équivaut à 'dwi', détruis un mot et entre en mode d'insertion.

r remplace le caractère sous le curseur par le caractère tapé après le 'r'.

~ (tilde) change une majuscule en minuscule ou le contraire.

Ces commandes sont particulièrement utiles quand on définit ses touches fonctions, ou quand on écrit des macros (séquences de commandes que lon enregistre en éditant, et qu'apres on peut rappeller autant de fois que c'est nécessaire).

Évidemment, là nous n'avons meme pas égratiné les possiblitiés de vi, qui prend réellement son envol, quand on intègre des commandes shell telles que grep, ctags et qu'on en connait un 60aine de commandes internes...

2.3 Visualisation des ressources système de base

2.3.1 Combien de place prend mon arborescence de fichiers ?

Pour savoir cela existe la commande **du** (comme disk usage). Celle-ci sans paramètre donnera comme résultat une liste avec la taille des sous répertoires et le nom du sous répertoire. La taille du répertoire courant se trouvera en fin de liste à côté du '.' (symbole pour le répertoire courant).

Pour avoir les tailles en mode "humainement lisible" attachez l'option '-h'.

2.3.2 Combien reste-t'il de place sur mes disques ?

La commande **df** (comme disk free) donne la taille des partitions montées (1.11.1) ainsi que leur taux de remplissage. Cela nous donne aussi une idée de ou se trouvent nos données et ce que nous utilisons comme supports de stockages.

Pour avoir les tailles en mode "humainement lisible" attachez l'option '-h'.

2.3.3 Combien reste t'il de mémoire libre ?

Au cas ou vous rencontreriez des problèmes d'allocation de mémoire, ou si vous voulez savoir si un processus que vous allez lancer aura des chances de tourner dans la mémoire centrale, il y a la commande **free**. Celle-ci vous retournera comme résultat l'étendue de la mémoire centrale, ainsi que celle de l'espace swap et leur taux d'utilisation.

Pour avoir les tailles en mode "humainement lisible" attachez l'option '-h'.

Chapitre 3

Facilités de BASH

'bash' offre certains services au niveau de l'entrée, qui vous faciliteront considérablement son utilisation.

- Les flèches horizontales vous permettent de corriger ce que vous avez tapé. Avec les flèches verticales, vous évoluez dans 'l'historique' du shell, c'est à dire vous avez accès à toutes les commandes que vous avez tapées depuis l'instantiation (la création) du shell.
- La touche <TAB> vous complète les noms de commandes ou de fichiers. Quand le tronçon de nom ne suffit pas à identifier le nom demandé, un signal retentit. Avec <SHIFT><TAB> peuvent être affichés tous les choix possibles.
- les 'wildcards', pour désigner des groupes de fichiers, il y a entre autres '*' qui remplace 0 ou plus caractères (ex. ls toto* affichera tous les fichiers qui commencent par toto, ls toto*.c affichera tous les fichiers commençant par toto et se terminant par '.c') et '?' qui remplacera un caractère exactement (ex. ls toto?.c affichera toto1.c mais pas toto11.c).
- '!'. Répétition de commandes : '!!' a le même effet que la flèche vers le haut, rappelle la dernière commande. '!<début de la commande> rappelle la dernière commande commençant par le tronçon que vous avez donné.
- Il arrivera que vous voudriez savoir quel exécutable est lancé, ou bien connaître sa location. 'which <nom_de_la_commande>' vous donnera le chemin ('path') exact de l'application. Exemple :

```
09:30:45 doc:~$ which bash
/usr/bin/bash
```

3.0.1 La re-direction

- Souvent apparaît la nécessité de re-diriger le flux de données qui vient sur la console dans un fichier, pour le traiter plus aisément par la suite. Pour cela il y a les commandes '>' et '>>'. La première crée un fichier et y inscrit le flot, en écrasant un éventuel fichier de même nom. La deuxième commande rajoute le flot à un fichier existant ou en crée un s'il venait à manquer. Exemple

`cat fichier1 fichier2 > somme_des_fichiers`
produit un fichier somme_des_fichiers qui contient les deux fichiers d'entrée.

Le tube (angl. the pipe) . Autre cas de re-direction, vous voulez que le flux de données serve d'entrée pour une autre application. Ici s'applique ce qu'on appelle 'tube ou pipe : '|'. Exemple
`ls | less`
Re dirige le flux de données de 'ls' dans 'less', permettant ainsi une lecture facile d'un répertoire avec beaucoup d'entrées.

3.0.2 Fonctionnalités de BASH

En outre de l'évolution dans l'arborescence d'un système de fichiers et le lancement d'exécutables, un bon shell vous donne plus de fonctions pour vous faciliter la vie :

variables

Dans tout shell qui se respecte, on trouve des variables. Il y a les variables dédiées au système ou à certains logiciels, qui en permettent la configuration au vol. Pour voir ce qu'il existe déjà comme variables vous pouvez taper la commande `set` sans argument, pour une meilleure visibilité une re-direction dans un fichier ou un "pager" est conseillée (p.ex. 'set | less').

L'assignement prend la forme suivante :

```
NOM=valeur_de_la_variable
```

La valeur est récupérée en utilisant le nom de la variable précédé par un '$'. Ainsi donnera

```
echo $NOM
valeur_de_la_variable
```

A noter que cet assignement de variable n'est valide que dans l'environnement dans lequel la variable a été définie. Ainsi une variable définie dans un shell script sera valide dans l'ensemble de celui-ci, mais pas à l'extérieur, de même chaque ligne que vous tapez sur le shell est un environnement séparé et clos.

Donc si vous voulez changer ou rajouter globalement une variable vous devrez l'**exporter** ' dans l'environnement, par exemple si vous voulez lire des 'News' et que votre machine ne connaît pas de serveur, ou que le serveur par défaut ne vous plaît pas vous taperez

```
export NNTPSERVER=news.u-strasbg.fr
```

Le fait de pouvoir mettre l'export sur la même ligne que l'assignement est une facilité bash et incompatible avec la sémantique sh. Dans la sémantique sh on déclare sa variable et sur une autre ligne on l'exporte.

Quelques variables intéressantes :

— `NNTPSERVER=yoda.u-strasbg.fr` pour pointer sur le bon serveur de news (p.ex. yoda.u-strasbg.fr)
— `FIGNORE=.o:.dvi:.class` pour ignorer par l'expansion de nom de fichier certains suffixes. Dans ce cas en présence de deux fichiers javaTest.java et javaTest.class, en tapant j<TAB> on obtiendra tout de suite javaTest.java s'il n'y a pas d'autre fichier qui commence par 'j'.
— Le Prompt `PS1`
Le prompt est la chaîne de caractères qui apparaît à chaque nouvelle ligne et qui normalement se termine par un '$' ou un '>'. Du style : `14:32:55 yoda: /tex$`. Cette chaîne peut être mise au goût de l'utilisateur à travers la variable shell PS1. Par exemple pour avoir un prompt comme dans l'exemple ci-dessus, il faudrait rajouter dans son .bashrc une ligne du style
`export PS1="\t \h:\w$ "`
Pour un complément d'information en ce qui concerne les possibilitées offertes dans le prompt se référer à la man-page de bash dans la section "PROMPTING".
— `TMOUT=3600` pour auto délogguer (se déconnecter) du shell après 3600 secondes.

Les boucles

La fonction 'for'. Si vous devez appliquer répétitivement une manipulation sur des fichiers, au lieu de le faire à la main, utilisez une boucle 'for'. Par exemple, vous avez des sources de programme où vous voudriez rajouter un en-tête. Mettez l'en-tête dans un fichier, supposons que vos sources aient toutes l'extension .C. Dans ce cas la commande shell

```
for i in *.C; do cat en_tete $i > $i; done
```

fera l'affaire. Dans la même rubrique vous avez des commandes

if [expression] then liste_de_commandes else fi extrait de man bash ... concernant les expressions possibles dans le **if**
— while list do list done
— until list do list done
— [function] name () list ;

Les scripts

Pour leur utilisation exacte référez vous à 'man bash'. Toutes ces commandes peuvent être dans un fichier qui aura comme première ligne

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

et sera déclaré exécutable (chmod). Cette sorte de fichier s'appelle un **script**.

Les commandes qui suivent sont en gros les mêmes que celles que vous tapez sur le shell à la main, donc vous pouvez tester votre séquence à la main sur la shell et la transcrire ensuite dans un script. À savoir que les variables déclarées dans le script n'ont pas besoin d'être exportées, elles sont valables pour toute la durée du script. La raison est simple : pour chaque commande que vous tapez, en fait le shell invoque un nouveau shell pour traiter cette ligne, shell qui, en gros, est initialisé par l'environnement du shell qui l'a généré, alors que le script est traité par un seul shell.

Autre chose, assez vite vous voudrez faire des scripts qui prennent des arguments, ceux ci sont stockés dans des variables du genre $1 $2 etc (au nombre des arguments livrés).

3.0.3 Adaptation personnelle de BASH

L'initialisation du shell se fait grâce aux fichiers : '/etc/profile' et '.bashrc', '.bash_profile','.bash_login' et '.bash_logout' dans votre répertoire.

Dans le fichier /etc/profile se trouvent toutes les initialisations de base, valables pour tous les utilisateurs. Ce fichier permet de faire tourner un shell sans le recours d'un fichier '.bashrc'. La syntaxe utilisée dans les deux fichiers est la même, vous êtes encouragés à visualiser le fichier /etc/profile pour apprendre la syntaxe utilisée et prendre connaissance des alias prédéfinis.

Dans .bash_profile vous avez des variables à déclarer lors de la première invocation d'un shell pour vous, et qui ne bougeront plus par la suite.

.bash_login est le fichier de configuration qui est lancé par un shell dit de "login", donc classiquement le shell où vous vous trouvez après le login. Pourquoi cette différence ? Eh bien, on n'a pas forcément besoin des mêmes réglages pour un shell que l'on veut utiliser manuellement (le shell de login précisément) et les shell classiquement invoqués pour des scripts.

.bash_logout finalement est ce qui est exécuté quand vous quittez un shell, dans la majorité des cas ce fichier ne contiendra que 'clear' dans un souci de discrétion.

3.1 Conclusion

Maintenant que vous avez vu la manipulation de base d'un shell, vous ne savez toujours pas ce qu'on peut faire avec UN*X, je vous conseille donc de compulser aussi mon guide aux utilitaires de base.