Mini aide-mémoire Unix

1. Ci-dessous quelques commandes de base. Dans la syntaxe, les crochets indiquent des arguments facultatifs. Pour une description plus complète, utiliser `man <nom_de_commande>`.
Pour l'instant, **sortie standard** = écran, **entrée standard** = clavier

cat : syntaxe `cat [-options] fic1 fic2…ficn`. Écrit sur la sortie standard les fichiers `fic1 fic2…ficn` dans cet ordre. Sans argument, écrit l'entrée standard sur la sortie standard (fin de fichier = ^D). Principales options : -n (numérote les lignes) -b (idem, mais les lignes blanches ne sont pas numérotées) -s (fusionne les lignes blanches consécutives) -v, -e ou -t (affichage plus ou moins complet des caractères non-imprimables).
cc : nom standard du compilateur c. Syntaxe élémentaire :
`cc -o nom fic1.c …ficn.c -lxxx -lyyy…`
fic1.c …ficn.c sont passés au préprocesseur cpp, compilés séparément pour générer des .o (objets) qui sont liés pour générer un exécutable appelé nom (a.out si il n'y a pas d'option -o nom). L'éditeur de liens utilise les bibliothèques xxx, yyy.
cd : Syntaxe `cd catalogue`. L'argument par défaut est le catalogue de login ("home directory"), `~user` est le catalogue de login de l'utilisateur `user`, `~` le votre. cd – ramène au catalogue précédent (bascule).
chmod : change les permissions. Syntaxe : `chmod b[+|-]p ficnom` ajoute (avec +) ou supprime (avec -) la permission p (p = r, w , x, s, t…) pour le bénéficiaire b (b = u, g, o, a) au fichier ficnom.
cmp : compare. Syntaxe : `cmp fic1 fic2`. Les 2 fichiers sont comparés ; le premier caractère différent est affiché avec son numéro.
cp : copie. Syntaxe : `cp fic1 fic2` copie fic1 sous le nom fic2. `cp fic1 fic2…ficn dirnom` copie les fichiers fic1 fic2…ficn dans le catalogue dirnom. Options : -i (interactive : demande confirmation avant d'écraser un fichier).
echo : Syntaxe `echo [-n] arg1 … argn`. Écho à l'écran des arguments, tels qu'ils sont évalués par le shell. L'option -n supprime le retour chariot en fin de ligne.
emacs : éditeur programmable libre maintenu par GNU. Un instrument de travail à apprendre peu à peu. Particularité : a été porté aussi sous Windows.
file : syntaxe `file fic1…ficn`. Essaye de déterminer le type des fichiers fic1,… ficn. (la liste exhaustive des types est donnée par man 4 file)
finger syntaxe `finger user`. Affiche les renseignements connus du système sur l'utilisateur user.
gcc : compilateur C de GNU. Gratuit. Syntaxe élémentaire : `gcc fic1.Φ…ficn.Φ`, où Φ peut être c, C, cc, s, o (fichier C, Cplusplus, assembleur, objet). Pour le reste, voir cc. Particularité : a été porté aussi sous Windows.
head : syntaxe `head [-n] [fichier1…fichiern]`. affiche les n premières lignes de chaque fichier (10 lignes par défaut, l'entrée standard si aucun nom de fichier n'est donné)
hostname : écrit sur la sortie standard le nom de la machine hôte
less : une version améliorée de more
ls : liste les fichiers. Syntaxe : `ls [-aldF…] arg1 … argn`. Affiche pour chaque argument : si c'est un fichier non catalogue, sa description ; si c'est un catalogue, la description de son contenu. Principales options : -a (affiche aussi les fichiers .xxx), -l (format

long, c'est à dire détaillé), -d (renseignements sur les catalogues eux même au lieu de leur contenu), -F (format court avec indication du type de fichier)
make : utilitaire de développement et de maintenance. Syntaxe élémentaire : `make [but]`. **Si `but` suivi de .c, .cc, .s ou .o est le nom d'un fichier, *make* génère la commande de compilation qui produit `but`**. Sinon, *make* cherche but dans le fichier `makefile` ou `Makefile` dans le catalogue courant et exécute les commandes correspondantes.
`makefile` a la forme suivante (simplifiée) :

```
but :     condition
      commande1
      commande2
      …
```

Par défaut, make utilise le premier but dans ce fichier.
man : le manuel : documentation en ligne sur les commandes, le langage C… Syntaxe : `man [section] titre` ou `man -k motclef` ou `man -f fichier`. Affiche les pages du manuel portant sur titre, ou indique celles qui renvoient à *motclef* ou a *fichier*.
mkdir : crée un catalogue. Syntaxe `mkdir chemin`. Chemin peut être relatif ou absolu.
more : syntaxe : `more [-options] fic1…ficn`. Affiche le contenu des fichiers passés en argument, en stoppant à chaque écran. Possède une aide intégrée (tapez h), décrivant diverses commandes dont une fonction de recherche.
mv : déplace (d'un catalogue à un autre) ou renomme (dans le même catalogue) un fichier. Syntaxe et options : cf cp.
pwd : écrit sur la sortie standard le nom du catalogue courant
rm : syntaxe `rm [-ri…] fic1…ficn`. Détruit les fichiers (ou catalogues) fic1,…ficn. Options : -r (récursif, c'est à dire détruit aussi les catalogues et leur contenu), -i (interactif, c'est à dire avec demande de confirmation)
sort : syntaxe : `sort [-options] [+pos1 [-pos2]]… fic1…ficn`. Fusionne et trie les lignes des fichiers fic1…ficn, écrit le résultat sur la sortie standard. Le tri est fait sur le contenu des lignes entre pos1 et pos2. L'option -u fusionne les listes identiques.
tail : syntaxe : `tail [+n | -n ] fichier` Ecrit sur la sortie standard la fin de fichier, à partir de la ligne numéro n (+n) ou les n dernières lignes (-n). Si on accole r au nombre n, inverse l'ordre des lignes.
tr : syntaxe `tr [-opt] chaine1 chaine2`. Recopie l'entrée standard sur la sortie standard en remplaçant tout caractère de chaine1 par le caractère de position correspondante dans chaine2.
wc : syntaxe : `wc [-opt] fichier`. Compte les éléments indiqués par opt, qui est une combinaison de c (caractères), w (mots), l (lignes).
whereis : syntaxe : `whereis fichier`. Cherche tous les exemplaires de fichier (en principe un exécutable) qui sont dans un catalogue du PATH
which : syntaxe : `which fichier`. Indique quel exemplaire de fichier sera lancé quand on tape la commande de nom "fichier"
who : syntaxe : `who`. Écrit sur la sortie standard la liste des utilisateurs actuellement logués. Avec l'argument `am i` ou `i am`, fournit l'identité de l'utilisateur.

2. Raccourcis clavier dans un terminal X (ou une console)
Quand ALT ne marche pas, essayer ESC (attention: on tape ESC puis *la touche indiquée) ...*
Flèche haut/bas ou ^p/^n pour parcourir l'historique des commandes, ^a/^e : début /fin de ligne, ALTb/ALTf : un mot en arriere / en avant, ALT_ : réécrit le dernier mot de la ligne précédente.
TAB : complète les noms de commande ou de fichier.