SCIENCES SORBONNE UNIVERSITÉ

Une Anti-sèche recto-verso pour GNU/LinuX

Les mots en **gras** ont valeur d'exemple, ceux en *italique* sont génériques.

GNU : un O.S. sous licence libre (acronyme récursif de **G**NU's **N**ot **U**nix)

Raccourcis clavier

au niveau du gestionnaire de fenêtres :

`Ctrl Alt T`	Ouvre une console (un ***t**erminal*) ⇔ `xterm &`
`Alt Tab`	Change de fenêtre active (Tab = ↹)
`Alt F4`	Ferme la fenêtre active
`Ctrl Alt` ⇄	Change de bureau virtuel
`Win` ⇄	Maximise la fenêtre sur un demi écran

dans la console (l'invite de commandes) :

↑↓	Navigue dans l'historique des commandes
`Ctrl A`	Place le curseur en début de ligne
`Ctrl E`	Place le curseur en fin de ligne
`Ctrl` ⇄	Se déplacer d'un mot
`Ctrl K`	Efface la fin de la ligne, après le curseur
`Ctrl U`	Efface le début de ligne, avant le curseur
`Ctrl W`	Efface le mot précédant le curseur
`Ctrl Y`	Colle le texte effacé (***y**ank*)
`Ctrl L`	Efface le contenu de la console ⇔ `clear`
`Ctrl` ±	Agrandit/réduit la police des caractères
`Ctrl C`	Termine le processus en cours ⚠≠copier (***c**lose*)
`Ctrl Z`	Suspend le processus en cours (c.f. `fg`)
`Ctrl D`	Fin de fichier (fin de saisie clavier, ou *idem* `exit`)

Arborescence des dossiers

`/`	La racine du système de fichiers
`~`	Alias de votre dossier d'accueil (i.e. `/home/jef`)
`.`	Le dossier de travail actuel (*working directory*)
`..`	Le dossier parent (`cd ..` pour remonter d'un niveau)
`/home/jef/Tux`	Adresse **absolue** du dossier **Tux**
`./Tux` ou `Tux`	Adresse **relative**, depuis le dossier actuel

Tout fichier ou dossier dont le nom commence par **.** est **caché**
L'extension est indicative. Les noms sont sensibles à la casse (MAJ./min.)

commandes du Noyau (de l'interpréteur, du *shell*)

Syntaxe d'invocation usuelle : ***cmd -options arguments***

♡	`ls`	Affiche le contenu du dossier de travail (***l**ist*)
	`ls Tux`	Affiche le contenu du sous-dossier **Tux**
	`ls -alh`	Avec fichiers cachés (***a**ll*) et détails (***l**ong*), lisibles par l'**h**umain (taille en `ko`, `Mo`, `Go`)
	`ls -ltr`	Dans l'ordre chronologique (*time*) inverse (*reverse*)
♡	`mkdir Tux`	Créer un dossier **Tux** (***m**ake **d**irectory*)
	`pwd`	Affiche le dossier de travail (***p**rint **w**orking **d**ir.*)
♡	`cd Tux`	Se placer dans le sous-dossier **Tux** (***c**hange **d**ir.*)
	`cd`	Se placer dans son dossier d'accueil ⇔ `cd ~`
	`cd -`	Revenir au dossier de travail précédent
♡	`cp ga bu`	Copier le fichier **ga** vers **bu** (***c**opy*)
	`cp ga bu zo Tux`	Copie 3 fichiers dans le dossier **Tux**
♡	`mv ga bu`	Déplace (renomme) **ga** vers **bu** (***m**ove*)
	`mv -v bu zo Tux`	Déplace 2 fichiers dans **Tux** (**v**erbeux)
♡	`rm ga`	Supprime le fichier **ga** définitivement (***r**emove*)
	`rm -rf Tux`	Supprime le dossier **Tux** (recursif et **f**orcé) ☠
♡	`rmdir Tux`	Supprime le dossier **Tux**, s'il est vide
	`ln -s ga lien`	Crée un lien **s**ymbolique vers le fichier **ga**
	`cmd --help`	Affiche l'aide pour la commande *cmd*
♡	`man` *cmd*	Le manuel † de la commande
	`history`	Historique des commandes précédentes
	`exit`	Ferme la console
	`lpr -P`*imp* **d.pdf**	Imprime **d.pdf** sur *imp* (***l**ine **p**rinter **r**emote*)
	`lpq -P`*imp*	La **q**ueue pour l'imprimante *imp*
	`lprm -P`*imp id*	Retire la tâche *id* de la queue d'impression

Copie à la souris et Auto-complétion

Le dernier texte surligné peut être collé avec le bonton central de la souris
Ceci indépendemment du texte coupé par `Ctrl-K` ou copié par `Ctrl-C`

`Tab`	Complète la commande ou l'adresse rapide et infaillible
`Tab Tab`	Affiche les possibilitées, si complétion ambigüe

L'abus de tabulation est instamment recommandé

Contenu des fichiers texte (ASCII)

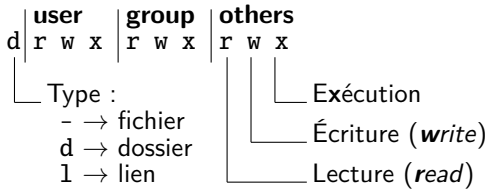
`more` **fic**	Affiche dans la console le contenu du fichier †
`less` **fic**	*idem*, mais sans persistance de l'affichage †
`head` **fic**	Affiche les premières lignes (`-n` pour choisir combien)
`tail` **fic**	Les dernières lignes (`-n +5` : à partir de la 5ième)
`diff` **ga bu**	Liste les différences entre **ga** et **bu**
`wc` **fic**	Les nombres de lignes, de mots et de caractères

Manipulation de fichiers

`cat` **ga bu zo**	Con**cat**enation des contenus, affichés à l'écran
`nano` **ga**	Éditeur de fichier texte, dans la console
`join` **a b**	Réunit les colonnes de **a** et **b** selon une, commune
`touch` **ga**	Création du fichier **ga**, ou change sa date de modif.

Permissions : lecture, écriture et exécution

La commande **`ls -l`** affiche en première colonne les droits associés au fichier : pour le propriétaire (***u**ser*), pour les utilisateurs du même **g**roupe, et pour les autres (***o**thers*) :



`chmod` *mode* **ga**	Modifie les permissions de **ga** selon le *mode* choisi, au format `[ugoa][+-=][rwx]`

Par exemple : **`ug+x`** donne au propriétaire et à son groupe le droit d'exécution (ou de traverser le dossier) ; **`o-wx`** retire aux autres les droit d'écriture et d'exécution ; **`a=r`** ne donne à tous (***a**ll*) que le droit de lecture

caractères Génériques `\␣*?#[]{}();,'"` `!&$<>|`

Pour empêcher leur interpolation, ils doivent être protégés par un \
L'espace ␣ est le séparateur d'arguments des commandes. **Enter** les valide

`*`	S'interpole en toute suite de caractères y compris vide (joker)
`?`	Un caractère quelconque, et un seul
`#`	Ce qui suit un # est ignoré (mis en commentaire)
`$`	Introduit une variable d'environnement
`;`	Séparateur de commandes
`[iou]`	Un caractère parmi l'ensemble (3 voyelles ici)
`[a-z] [A-Z]`	N'importe quelle lettre minuscule majuscule
`[a-zA-Z]`	N'importe quelle lettre
`[0-9]`	N'importe quel chiffre
`{bu,zo,meu}`	L'un des mots de la liste
`` `cmd` ``	Remplacé par l'évaluation de *cmd* ⇔ `$(`*cmd*`)`
`'expr'`	Protège une expression ne contenant pas de **'**
`"expr"`	Protège l'expression, mais honore les **$ \ `**
`&&` `||`	Opérateurs logiques **et** **ou**
`!␣`*cmd*	Complément logique de la valeur de retour de *cmd* (*not*)
`!x`	Rappel de la dernière commande commençant par **x**

Par exemple, **`*.*`** correspond aux fichiers ayant une extension
On recense aussi **`=`** , **`[[ ]]`** et **`(( ))`** (*c.f.* **`man bash`**)
Ces caractères sont à proscrire dans les noms d'entitées ⇒ **_** au lieu de ␣

†. **Espace** pour la page suivante, **q** pour quitter, ***/motif*** pour rechercher un motif (**n**/**p** pour aller à l'occurence suivante/précédente)

▷ lancer une tâche en arrière-plan `&`

La commande s'exécute au 1er plan, et bloque la console avant de terminer

cmd `&`	Exécute la commande *cmd* en arrière-plan
`bg`	Relance en arrière-plan la tâche interrompue (par `Ctrl-Z`)
`fg`	Ramène au 1er plan la tâche d'arrière-plan (*i.e.* avant `Ctrl-C`)

Par exemple : `firefox ; Ctrl Z ; bg` ⇔ `firefox &`

▷ parmi les variables d'environnement

Elles sont introduites par **$** ou **${...}**. Utiliser **echo** pour les afficher

`$HOME`	Le dossier d'accueil ⇔ ∼
`$USER`	Le *login* de l'utilisateur (*i.e.* `jef`)
`$HOSTNAME`	Le nom de la machine utilisée
`$SHELL`	L'interpréteur de commande (le noyau `bash`, `sh` ...)
`$PATH`	Les dossiers de recherche des exécutables
`$?`	Valeur de retour (= code d'err.) de la dernière commande

▷ Flux de sortie *stdout* et Redirection `>`

cmd `>` **fic**	Redirection vers **fic** du flux de sortie de la *cmd*
cmd `>>` **fic**	En ajout : flux de sortie ajouté à la fin de **fic**
cmd `|` `tee` **fic**	Duplique (⊤) le flux de sortie vers **fic**
cmd `2>` **err**	Redirige la sortie d'erreur (2) vers le fichier **err**
cmd `2>>` **err**	Ajout des erreurs à la fin du fichier **err**
cmd `2>&1`	Réunit erreurs et sortie standard (1) avant |
cmd `>`**fic** `2>&1`	Erreurs et sortie standard vers **fic** ⚠ ordre

La redir° entrante ***cmd*<in** remplace la saisie au clavier par le contenu de **in**
`echo "ligne 1">`**f**`; echo "ligne 2">>`**f** produit un fichier **f** de 2 lignes

les Filtres et le Tube `|`

Les filtres s'appliquent à un flux entrant : `cat` **ga** `|sort`
ou à un fichier : `sort` **ga** (sauf le filtre pur **tr**)

cmdX `|` *cmdY*	Utilise la sortie de *cmdX* en entrée de *cmdY*
`sort -k3,3nr`	Ordonne **n**umériquement, en ordre inv**r**se, selon la 3ième colonne
`uniq`	Supprime les lignes consécutives identiques
`grep -n` *motif*	Les lignes, **n**umérotées, contenant le *motif*
`grep -v` *motif*	Les lignes ne le contenant pas (**v**eto)
`grep -C 3` *mtf*	Affiche 3 lignes de **c**ontexte avant et après
`sed 's/`**mot**`/`**alt**`/g'`	**S**ubstitue ‡ **g**lobalement **mot** par **alt**
`sed '3,7d'`	Supprime (***d***elete) les lignes 3 à 7 (***s***tream **ed***i*tor)
`tr 'iou' 'IOU'`	Translitteration lettre à lettre : 3 voyelles en maj.

i.e. `echo $PATH | tr ':' '\n'` ⇒ affiche un dossier du `PATH` par ligne

En l'absence de fichier ou de redirection entrante, les filtres travaillent sur la saisie clavier (*stdin*), qui se terminera avec **Ctrl-D**

les expressions Régulières *a.k.a. regexp* → grep, sed

`.`	S'interpole en un caractère quelconque
`*`	Plusieurs occurences du motif précédent, zéro inclu
`\`	Pour protèger ces 3 caractères spéciaux
`[iou]`	Un caractère parmis la liste
`[^iou]`	Un caractère **hormis** ceux de la liste (veto)
`^`	Ancre de début de ligne si placé au début du motif
`$`	Ancre de fin de ligne si placé à la fin du motif
`..*`	Au moins un caractère
`[^␣][^␣]*`	Un champ : ne contient pas d'espace ␣
`\(` *regexp* `\)`	Définit un groupe (`\`***i*** pour s'y référer ensuite)

i.e. `sed 's+^␣*\([^␣][^␣]*\)␣␣*\([^␣][^␣]*\)+\2␣\1+'`
permute les deux 1er champs de chaque ligne (exempte de tabulat°)

Il existe une variante **étendue** des *regexp* (`grep -E`; `sed -E`; `awk`)

Configuration et Administration du système

`su`	Prendre le rôle de super-utilisateur (*admin.*)
`sudo` *cmd*	Exécute la *cmd* en tant que super-user
`dmesg`	Les message du noyau depuis le démarrage
`∼/.bashrc`	Le fichier de configuration du noyau
`∼/.bash_profil`	Configurations au démarrage (au *login*)
`/etc/fstab`	Table de partition des disques
`/proc/cpuinfo`	Informations sur les microprocesseurs

quelques commandes utiles

`env`	Affiche les variables d'environnement
`top`	Le palmarès des processus § (charge CPU ou RAM)
`ps -u` **jef**	Les processus lancés par **jef** et leurs *Id*
`kill -9` *jobId*	Tue le processus n° *jobId* (`-9` s'il résiste)
`killall` **python**	Tue tous les processus **python**
`echo` *expr*	L'expression *expr*, interpolée par le noyau
`file` **fic**	Le type du fichier **fic** (texte, binaire, pdf...)
`which` *cmd*	L'adresse du fichier exécutable *cmd*
`du -s` **Tux**	Espace utilisé par **Tux** (***d***isk ***u***sage, ***s***ummary)
`df -h`	Utilisation des partitions (***d***isk ***f***ree)
`find` **Tux** `-name "*.c"`	Les fichiers d'extension **c** sous **Tux**
`find` **Tux** `-size +10M`	Les fichiers de plus de 10 Mo
`gzip` **ga**	Compresse le fichier **ga** ⇒ **ga.gz**
`gunzip` **ga.gz**	décompresse **ga.gz**
`tar czf` **a.tgz Tux**	**C**rée l'archive **a.tgz** de **Tux**, **z**ippée
`tar tzf` **a.tgz**	**T**este le contenu de l'archive
`tar xzf` **a.tgz**	E**x**trait l'archive dans le dossier actuel
`make` *cible*	Compile la *cible* selon la recette du fic. **Makefile** ...

les Raccourcis de commande

`alias`	Affiche la liste les alias déjà définis
`alias` **ll**	La définition de l'alias **ll**, s'il existe
`alias` **lt**`='ls -ltr'`	Définit un alias **lt**
`\`*cmd*	La commande primitive (ignore l'alias)

Internet et serveurs

`wget` *url*	Copie la page web trouvée à l'*url*
`ping` *srvr*	Teste le chemin vers un serveur
`ssh` **jef**`@`*srvr*	Connexion à un serveur (***s***ecure ***sh***ell)
`scp` **ga** **jef**`@`*srvr*`:`**Tux**	Copie **ga** vers le dossier **Tux** de *srvr*
`logout`	Ferme la connexion à distance

pour l'affichage à distance de fenêtres graphiques, ajouter à **ssh** l'option **-Y**

le filtre programmable AWK

Syntaxe : **awk '***condition***{***actions***}' *fichier***

Les actions sont appliquées à chaque ligne du fichier
$k est le $k^{\text{ième}}$ champ ($1 \leq k \leq$ `NF`), et **$0** l'ensemble de la ligne
NR et **NF** : n° de ligne (***r***ecord) et nombre de colonnes (***f***ields)
FS et **OFS** : séparateurs de champs de l'entrée, et en sortie ...

`awk '{print NR, $0}'` **fic**
ajoute le numéro de ligne en première colonne
`awk '{c=$2; $2=$1; $1=c; print $0}'` **fic**
permute les deux premières colonnes de **fic**
`awk 'NR>=4 && NR<10 && NF>1'` **fic**
les lignes 4 à 9 ayant 2 champs ou plus (action implicite : `print $0`)
`awk 'BEGIN{s=0}{s += $1}END{print s/NR}'` **fic**
moyenne de la 1ère colonne (`BEGIN` superflu : variables initialisées à `0`)
`awk '{s=0; for(j=1;j<=NF;j++) s+=$j; print s/NF}'` **f**
calcule la moyenne de chaque ligne du fichier **f**
`awk 'BEGIN{FS=","; OFS="\t"}{print $(NF-1),$NF}'` **f.csv**
les 2 derniers champs séparés par **Tab**, d'un tableau séparé par des virgules
`awk -v moi=${USER} '{print moi, $0}'` **fic**
import de variable d'env. : ajoute le *login* en début de ligne
`cat` **fic** `| awk '$1>0'`
filtre les lignes dont le 1er champ est positif (sur le flux entrant)

Programmer : la puissance du *shell script*

Commandes dans un fichier (*i.e.* `scrpt.sh`) ⇒ interprétées à l'exécution (`bash scrpt.sh [args]`). **$*i*** est le $i^{\text{ème}}$ argument, **$#** leur nombre ...

```
for fic in *.png ; do
  out=${fic%.png}.jpg
  convert "$fic" "$out"
done
```
png2jpg.sh

Convertit en JPG tous les fichiers PNG du dossier de travail
(fragile : si aucun `PNG` n'est présent)

‡. Tout caractère peut servir de séparateur à la place de **/** , *i.e.* `sed 's+mot+alt+g'`

§. **q** pour quitter. **M** pour classer selon la RAM, **T** selon la durée, **P** pour revenir au CPU, etc.