

Bash - Edition

<code>^k</code>	Coupe du curseur jusqu'à la fin de ligne
<code>^u</code>	Coupe du curseur jusqu'en début de ligne
<code>^w</code>	Coupe le mot avant le curseur
<code>^y</code>	Coller une chaîne précédemment coupée
<code>%Backspace</code>	Supprime un mot jusqu'à un symbole de type tiret...
<code>%d</code>	Supprime le mot suivant
<code>^h</code>	Remplace Backspace
<code>%c</code>	Met la première lettre en maj et avance d'un mot
<code>%u</code>	Met le mot en majuscule
<code>%l</code>	Met le mot en minuscule
<code>%. </code>	Réécrit le paramètre de la dernière commande
<code>%t</code>	Inverse la position des deux mots avant le curseur
<code>^t</code>	Inverse la position des deux caractères avant le curseur
<code>^</code> : Ctrl	
<code>%</code> : Alt	

Bash - Historique

<code>!!</code>	Relancer la dernière commande
<code>!p</code>	Relancer la dernière commande commençant par p
<code>!!:p</code>	Afficher la dernière commande commençant par l

Bash - Historique (cont)

<code>!\$</code>	Récupérer le dernier argument de la commande précédente
<code>!^</code>	Récupérer le premier argument de la commande précédente
<code>!*</code>	Tous les arguments de la dernière commande
<code>!*:p</code>	idem mais l'affiche
<code>!n</code>	Execute la nième commande
<code>history -c</code>	Vider l'historique
<code>!-2</code>	Execute la nième commande en partant de la fin
<code>!?pattern</code>	Execute la dernière commande contenant pattern
<code>pattern1 pattern2</code>	Remplace pattern1 par pattern2 dans la dernière commande

Bash - Divers

<code>%r</code>	Vide la ligne
<code>^r</code>	Recherche une commande déjà tapée
<code>^c</code>	Arrête la commande en court
<code>^d</code>	Quitte le shell en court
<code>^l</code>	Efface le contenu de l'écran
<code>^o</code>	Valide la ligne en cours
<code>tab</code>	Complétion
<code>%*</code>	Affiche les complétions disponibles

Bash - Variables

<code>\${var}</code>	Valeur de var
<code>\${var:-word}</code>	Affiche word si var est nulle ou unset
<code>\${var:=word}</code>	Affiche word si var est nulle ou unset et set assigne word à var
<code>\${var:?}</code>	Affiche une erreur si VAR est nulle ou unset
<code>\${var:+word}</code>	Affiche word si var est différente de nulle
<code>\${var:offset}</code>	Affiche var à partir de l'offset
<code>\${var:offset:length}</code>	Affiche var à partir de l'offset sur length de longueur
<code> \${!tab[@]}</code>	Liste les id du tableau tab
<code> \${tab[#_ID]}</code>	Affiche la valeur du #_ID de tab
<code> \${#var}</code>	Affiche la taille de var
<code> \${var#pattern}</code>	var amputé du pattern mini en prefix
<code> \${var##pattern}</code>	var amputé du pattern max en prefix
<code> \${var%pattern}</code>	var amputé du pattern mini en suffix
<code> \${var%%pattern}</code>	var amputé du pattern max en suffix
<code> \${var/pattern/string}</code>	Substitution



Bash - Variables (cont)

`${var^}`

Maj du premier caractère

`${var^^}`

Maj de tous les caractères

`${var,}`

Min du premier caractère

`${var,,}`

Min de tous les caractères

Bash - Deplacement

`^a` Aller en début de ligne

`^e` Aller en fin de ligne

`%b` Aller au mot précédent

`%f` Aller au mot suivant

`^xx` Alterne le curseur avec sa position précédente

`^p` Historique précédent

`^n` Historique suivant

`^` : Ctrl

`%` : Alt ou Esc

Bash - IO Redirections

`cmd > file`

Redirige stdout de cmd dans file

`cmd 2> file`

Redirige stderr de cmd dans file

`cmd &> file`

Redirige stdout et stderr de cmd dans file

`cmd < file`

Envoi le contenu de file dans cmd

`cmd 2> /dev/null`

Redirige stderr dans un trou noir

`cmd > file.out 2> file.err`

Redirige stdout dans file.out et stderr dans file.err

Bash - IO Redirections (cont)

`cmd1 | cmd2`

Redirection stdout de cmd1 dans stdin de cmd2. stderr n'est pas transmis dans les |

`cmd1 | cmd2 | cmd3 | cmd4; echo`

`${PIPESTATUS[@]}`

Suite de redirections et récupération des

> file

Vide et/ou crée un fichier

`cmd | tee cmd.out | sort | tee sort.out | uniq -c |`

`tee uniq.out`

Un fichier de sortie par cmd

`(cmd1; cmd2) > file`

stdout des 2 cmd dans file (via sous shell)

`{ cmd1; cmd2; } > file`

stdout des 2 cmd dans file (sans sous shell)

`cmd1; cmd2`

Execution cmd1 puis cmd2

`cmd1 && cmd2`

Execution de cmd2 si cmd1 est OK

`cmd1 || cmd2`

Execution de cmd2 si cmd1 est non OK

`>>` ajoute au lieu de rediriger.

