**BASH**

|  |  |
| --- | --- |
| **DESC** | |
| Bourne Again SHell | |
| **FILES** | |
| /etc/profile | Lu une fois (Configuration Shell de connexion) |
| ~/.profile (ou .bash\_profile) (ou .bash\_login) | Paramètres d’environnement Shell de connexion lu une fois   * Variables exportées/non exportées * Umask * Pas d’alias ni options * Lecture du fichier .bashrc |
| ~/.bash\_history | Historique cmd |
| ~/.bashrc | Options et alias |
| **QUICK START** | |
|  | |

**Généralités :**

* Shells :
  + Bourne Shell - sh (1970)
  + Bourne again Shell - bash
  + korn Shell - ksh (1983)
  + C Shell - csh
  + Variante C Shell - tcsh
* Les bits :
  + setUID :
  + setGID :
  + Sticky bit sur fichier :
  + Sticky bit sur un répertoire : tous les fichiers crées héritent automatiquement du proprio + groupe de celui du répertoire sticky
* Shell de connexion
  + Shell unique lancé juste après login/mdp en mode console
* Pour les expressions, c’est le Shell qui réalise la substitution et non la commande
* Un bash enfant est forké quand on lance une commande.
* . est un built-in du Shell et permet de lancer un programme dans ce Shell et non un descendant
* Les variables non exportées ne sont pas transmises aux descendants du Shell.
* [ est un lien symbolique vers "test" qui est aussi un built-in du Shell : d’où le fait qu’il faille des espaces dans : [ X ou Y ], [ prenant ses paramètres comme un programme normal.

|  |  |  |  |
| --- | --- | --- | --- |
| **DIVERS** | | | |
| ~ | Home directory  . désigne aussi le home directory | | |
| 'blabla' | Empêche l’interprétation des variables $ | | |
| "blabla" | Interprétation de $ " ` \  mais pas de > et | | | |
| \$ | Echappement d’un caractère | | |
| \* | Wildcard | | |
| **COMMANDES** | | | |
| Cmd & | Exécution en arrière-plan | | |
| bash script | **Exécution** du programme ou script par un **Shell enfant** | |  |
| ./prog | Doit être exécutable  ./ permet de spécifier le répertoire courant pour un programme |
| prog | Exécutable + se trouve dans le PATH |
| . ./prog | **Exécution** du programme **dans ce Shell** et non dans un descendant  . est un built-in du Shell | | |
| `blabla` | Interprétation d’une commande :  Fork du Shell puis run de la commande | | |
| $() | Idem | | |
| Cmd1 && cmd2 | Cmd2 executed if cmd1 pass | | |
| Cmd1||Cmd2 | Cmd2 executed if cmd1 fail | | |
| Cm1 ;cmd2 | Cmd2 will be executed even if cmd1 fail | | |
| (Cm1 ;cmd2) | Création d’un Shell enfant puis Shell enfant s’occupe des commandes  si on change de répertoire courant du coup à la fin des cmds, le fils meurt et le père garde le même répertoire qu’avant | | |
| { Cm1 ;cmd2 ; } | Attention ! Espace en début et en fin et ; à la dernière commande  ligne de commande traitée par le Shell courant et non plus un fils  Si redirections, les descripteurs sont sauvegardés puis restaurés en fin.  Accolades plus performante mais () ne modifient pas l’environnement du Shell courant + plus simple à utiliser donc préféré. | | |
| **VARIABLES** | | | |
| $VAR | Utilisation d’une variable | | |
| ${Opts} | Utilisation d’une variable dans " "  Permet de délimiter la variable | | |
|  | variable:-defVar | remplace par variable si existe et defVar sinon | |
|  | variable:=var | Idem ? | |
|  | variable:+var | si variable non vide, var est utilisée sinon affichage vide | |
|  | variable:?message | si variable existe, elle est utilisée sinon affichage du message d’erreur | |
|  | HOME,, | Convertit en minuscule | |
|  | #HOME | Longueur de la variable HOME | |
|  | variable#modele | Retire la chaine la plus courte en partant de la gauche | |
|  | variable##modele | Retire la chaine la plus grande en partant de la gauche | |
|  | variable%modele | Retire la chaine la plus courte en partant de la droite | |
|  | variable%%modele | Retire la chaine la plus grande en partant de la droite | |
| **REDIRECTIONS**   * **Cmd externe : Duplication du shell, positionne les descripteurs et exec()** * **Cmd interne : Positionne descripteurs, exécution cmd et MAJ des descripteurs d’avant** | | | |
| 0< | Redirige entrée standard | | |
| < |
| > | Redirige sortie standard. Création | Ecrasement. | | |
| 1> |
| 2> | Redirection sortie erreur | | |
| 2>&1 | Redirige stderr vers stdout  Surtout pas ls >1 fichier >2 fichier : offsets différents  Mais ls 1>fichier 2>&1 | | |
| >| | Forcing de la redirection (noclobber activé = impossible d’écraser) | | |
| >> | Idem mode append | | |
| < | Redirection entrée standard | | |
| **PIPE** | | | |
| Cmd1 | cmd2 | Pipe | | |
| **SIGNAUX – TOUCHES RAPIDES** | | | |
| CTRL + C | SIGINT : terminate the process by default | | |
| CTRL + D | EOF signal | | |
| CTRL + \ | SIGQUIT | | |
| CTRL + Z | SIGSTOP | | |

**Variables du Bash :**

|  |  |
| --- | --- |
| **Variable** | **Comments** |
| **GENERALE** | |
| $BASH\_VERSION |  |
| $HOME |  |
| $PWD |  |
| $PATH |  |
| $PS1 | Chaine de caractères de l’invite de commande |
| $PS2 | Idem prompt secondaire. |
| $PS3 | Prompt du menu select |
| $TMOUT | Delai en secondes avant extinction shell si aucune interaction.  0 = aucun timer |
| $LOGNAME | Nom du user connecté |
| $IFS | Séparateurs : par défaut espace, tabulation et saut de ligne |
| **PARAMETRES POSITIONNELS** | |
| $ ! | PID du dernier processus lancé en arrière-plan |
| $$ | PID Shell courant |
| $ ? | Dernière valeur de retour d’une commande Unix |
| **PARAMETRES POSITIONNELS PROGRAMME** | |
| $# | #Arguments (le nom n’est pas compté) |
| $0 | Nom du programme |
| $X | Argument n°X (de 1 à 9)  Pour accéder à des paramètres > 9, utiliser la commande shift |
| $\* | Représente tous les arguments  Attention : Le Shell enfant ne protège pas les guillemets :  a b c "d e f" g = 7 arguments |
| $@ | Idem sauf que le Shell enfant protège cette fois :  a b c "d e f" g = 5 arguments |

**Signaux :**

|  |  |
| --- | --- |
| **Signal** | **Comments** |
| SIGHUP | Fin de connexion |
| SIGINT | Interruption |
| SIGQUIT | Interruption forte |
| SIGKILL | Interruption très forte |
| SIGSTOP | Interruption temp |
| SIGCONT | Redémarrage |

**Expressions :**

|  |  |
| --- | --- |
| **Expression** | **Comments** |
| \* | Wildcard : représente tous les caractères |
| ? | Un caractère quelconque  Ex : ls ???? affiche tous les fichiers/dossiers de 4 caractères |
| [fo]  [a-zfb]  [!a-z] | f ou o  une minuscule ou f ou b  Tout sauf une minuscule |
| **Expressions étendues** | **Shopt –s extglob pour les activer sous Bash** |
| ?(expression) | Expression présente 0 ou 1 fois |
| \*(expression) | Présente entre 0 et n fois |
| +(expression) | Présente entre 1 et n fois |
| @(expression) | Exactement 1 fois |
| !(expression|expression…) | Ne sera pas présente |
| **REGEX** | **Comments** |
| ^ | Début de ligne |
| $ | Fin de ligne |
| . | Un caractère quelconque |
| [] | Liste de caractères possibles |
| [^] | Un caractère qui n’est pas dans cette liste |
| \* | Wildcard |
| \< | Début d’un mot |
| \> | Fin d’un mot |
| \c | Echappement de c |