Tableau 1. Variables spéciales du shell

Variable	*Signification*
`$0`	Nom du script
`$1`	Paramètre de position #1
`$2 - $9`	Paramètres de position #2 - #9
`${10}`	Paramètre de position #10
`$#`	Nombre de paramètres de position
`"$*"`	Tous les paramètres de position (en un seul mot) *
`"$@"`	Tous les paramètres de position (en des chaînes séparées)
`${#*}`	Nombre de paramètres sur la ligne de commande passés au script
`${#@}`	Nombre de paramètres sur la ligne de commande passés au script
`$?`	Code de retour
`$$`	Numéro d'identifiant du processus (PID) généré par le script
`$-`	Options passées au script (utilisant set)
`$_`	Dernier argument de la commande précédente
`$!`	Identifiant du processus (PID) du dernier job exécuté en tâche de fond

* Doit être entre guillemets, sinon il vaudra par défaut « `$@` ».

Tableau 2. Opérateurs de test : comparaison binaire

Comparaison arithmétique		*Comparaison de chaînes*	
`-eq`	Égal à	`=`	Égal à
		`==`	Égal à
`-ne`	Différent de	`!=`	Différent de
`-lt`	Plus petit que	`\<`	Plus petit que (ASCII) *
`-le`	Plus petit que ou égal à		
`-gt`	Plus grand que	`\>`	Plus grand que (ASCII) *
`-ge`	Plus grand que ou égal à		
		`-z`	Chaîne vide
		`-n`	Chaîne non vide

Entre des parenthèses doubles `(( ... ))`	
`>`	Plus grand que
`>=`	Plus grand que ou égal à
`<`	Plus petit que
`<=`	Plus petit que ou égal à

* Si à l'intérieur d'une construction de tests à double crochets `[[ ... ]]` , alors l'échappement \ n'est pas nécessaire.

Tableau 3. Opérateurs de test : fichiers

Opérateur	*Tests si*	*Opérateur*	*Tests si*
`-e`	Le fichier existe	`-s`	Le fichier est vide
`-f`	Le fichier est un fichier standard		
`-d`	Le fichier est un répertoire	`-r`	Le fichier a un droit de lecture
`-h`	Le fichier est un lien symbolique	`-w`	Le fichier a un droit en écriture
`-L`	Le fichier est un lien symbolique	`-x`	Le fichier a le droit d'exécution
`-b`	Le fichier est un périphérique bloc		
`-c`	Le fichier est un périphérique caractère	`-g`	L'option sgid est positionnée
`-p`	Le fichier est un tube	`-u`	L'option suid est positionnée
`-S`	Le fichier est un socket	`-k`	L'option « sticky bit » est positionnée
`-t`	Le fichier est associé à un terminal		
`-N`	Le fichier a été modifié depuis sa dernière lecture	`F1 -nt F2`	Le fichier F1 est plus récent que F2 *
`-O`	Vous êtes le propriétaire du fichier	`F1 -ot F2`	Le fichier F1 est plus ancien que F2 *
`-G`	L'identifiant du groupe du fichier est le même que vous	`F1 -ef F2`	Les fichiers F1 et F2 sont des liens vers le même fichier *
`!`	« NOT » (inverse le résultat des tests ci-dessus)		

* Opérateur binaire (nécessite deux opérandes).

Tableau 4. Substitution et expansion de paramètres

Expression	*Signification*
`${var}`	Valeur de var, identique à `$var`
`${var-DEFAUT}`	Si var n'est pas initialisé, évalue l'expression `$DEFAUT` *
`${var:-DEFAUT}`	Si var n'est pas initialisé ou est vide, évalue l'expression `$DEFAUT` *
`${var=DEFAUT}`	Si var n'est pas initialisé, évalue l'expression `$DEFAUT` *
`${var:=DEFAUT}`	
`${var+AUTRE}`	Si var est initialisé, évalue l'expression `$AUTRE`, sinon est une chaîne null
`${var:+AUTRE}`	
`${var?ERR_MSG}`	Si var n'est pas initialisé, affiche `$ERR_MSG` *
`${var:?ERR_MSG}`	
`${!varprefix*}`	Correspond à toutes les variables déclarées précédemment et commençant par `varprefix`
`${!varprefix@}`	

* Bien sûr, si var est initialisé, évalue l'expression comme `$var`.

Tableau 5. Opérations sur les chaînes

Expression	*Signification*
`${#chaîne}`	Longueur de `$chaîne`
`${chaîne:pos}`	Extrait la ss-chaîne à partir de `$chaîne` jusqu'à `$pos`
`${chaîne:pos:long}`	Extrait `$long` caractères dans la ss-chaîne à partir de `$chaîne` jusqu'à `$pos`
`${chaîne#ss-chaîne}`	Supprime la plus petite correspondance de `$ss-chaîne` à partir du début de `$chaîne`
`${chaîne##ss-chaîne}`	Supprime la plus grande correspondance de `$ss-chaîne` à partir du début de `$chaîne`
`${chaîne%ss-chaîne}`	Supprime la plus courte correspondance de `$ss-chaîne` à partir de la fin de `$chaîne`
`${chaîne%%ss-chaîne}`	Supprime la plus longue correspondance de `$ss-chaîne` à partir de la fin de `$chaîne`
`${chaîne/ss-chaîne/remplac}`	Remplace la première correspondance de `$ss-chaîne` avec `$remplac`
`${chaîne//ss-chaîne/remplac}`	Remplace toutes les correspondances de `$ss-chaîne` avec `$remplac`
`${chaîne/#ss-chaîne/remplac}`	Si `$ss-chaîne` correspond au début de `$chaîne`, substitue `$ss-chaîne` par `$remplac`
`${chaîne/%ss-chaîne/remplac}`	Si `$ss-chaîne` correspond à la fin de `$chaîne`, substitue `$ss-chaîne` par `$remplac`
`expr match "$chaîne" '$ss-chaîne'`	Longueur de `$ss-chaîne`* correspondant au début de `$chaîne`
`expr "$chaîne" : '$ss-chaîne'`	Longueur de `$ss-chaîne`* correspondant au début de `$chaîne`
`expr index "$chaîne" $ss-chaîne`	position numérique dans `$chaîne` du premier caractère correspondant dans `$ss-chaîne`
`expr substr $chaîne $pos $long`	Extrait `$long` caractères à partir de `$chaîne` commençant à `$pos`
`expr match "$chaîne" '\($ss-chaîne\)'`	Extrait `$ss-chaîne`* au début de `$chaîne`
`expr "$chaîne" : '\($ss-chaîne\)'`	Extrait `$ss-chaîne`* au début de `$chaîne`
`expr match "$chaîne" '.*\($ss-chaîne\)'`	Extrait `$ss-chaîne`* à la fin de `$chaîne`
`expr "$chaîne" : '.*\($ss-chaîne\)'`	Extrait `$ss-chaîne`* à la fin de `$chaîne`

* Où `$ss-chaîne` est une expression rationnelle.

Tableau 6. Constructions diverses

Expression	*Interprétation*
Crochets	
`if [ CONDITION ]`	Construction de tests
`if [[ CONDITION ]]`	Construction de tests étendue
`Tableau[1]=element1`	Initialisation d'un tableau
`[a-z]`	Ensemble de caractères se suivant à l'intérieur d'une expression rationnelle
Accolades	
`${variable}`	Substitution de paramètres
`${!variable}`	Référence de variable indirecte
`{ commande1; commande2; ... commandeN; }`	Bloc de code
`{chaîne1,string2,string3,...}`	Expansion
`{a..z}`	Expansion d'accolades
`{}`	Remplacement de texte, après find et xargs
Parenthèses	
`( commande1; commande2 )`	Groupe de commandes exécutées dans un sous-shell
`Tableau=(element1 element2 element3)`	Initialisation d'un tableau
`result=$(COMMANDE)`	Substitution de commande(s)
`>(COMMANDE)`	Substitution de processus
`<(COMMANDE)`	Substitution de processus
Double parenthèses	
`(( var = 78 ))`	Arithmétique entière
`var=$(( 20 + 5 ))`	Arithmétique entière, avec affectation de variables
`(( var++ ))`	Incrément de variables style C
`(( var-- ))`	Incrément de variables style C
`(( var0 = var1<98?9:21 ))`	Opération à trois arguments
Guillemets	
`"$variable"`	Guillemets « faibles »
`'chaîne'`	Guillemets « forts »