Opérateurs et caractères spéciaux en Bash

Ce tableau présente les opérateurs, les caractères spéciaux et les redirections couramment utilisés dans le shell Bash.

Caractère Opérateur	Nom	Description	Exemple
>	Redirection de sortie	Redirige la sortie standard vers un fichier (écrase le contenu existant)	ls > fichier.txt
>>	Redirection avec ajout	Redirige la sortie standard vers un fichier (ajoute à la fin)	echo "texte" >> fichier.txt
<	Redirection d'entrée	Redirige l'entrée standard depuis un fichier	sort < fichier.txt
2>	Redirection d'erreur	Redirige la sortie d'erreur vers un fichier	find / -name "*.conf" 2> erreurs.txt
2>&1	Redirection combinée	Redirige la sortie d'erreur vers la sortie standard	find / -name "*.conf" > resultats.txt 2>&1
&>	Redirection combinée	Redirige la sortie standard et d'erreur vers un fichier	ls /dossier_inexistant &> tout.txt
| (pipe)	Tube/Pipeline	Connecte la sortie d'une commande à l'entrée d'une autre	ps aux | grep apache
|| (double pipe)	OU logique	Exécute la commande de droite seulement si celle de gauche échoue (code de retour non-zéro)	ping -c1 google.com || echo "Pas de réseau"
&&	ET logique	Exécute la commande de droite seulement si celle de gauche réussit	make && make install
;	Séparateur de commandes	Exécute plusieurs commandes séquentiellement	ls ; echo "Terminé"
&	Exécution en arrière-plan	Lance une commande en arrière-plan	find / -name "*.log" &
*	Joker (wildcard)	Correspond à n'importe quelle chaîne de caractères	ls *.txt
?	Joker simple	Correspond à n'importe quel caractère unique	ls fichier?.txt
[]	Liste de caractères	Correspond à n'importe quel caractère dans la liste	ls [abc]*.txt
[!] ou [^]	Négation de liste	Correspond à n'importe quel caractère pas dans la liste	ls [!a-c]*.txt
{}	Expansion de chaînes	Génère plusieurs chaînes de caractères	echo fichier{1,2,3}.txt
()	Sous-shell	Exécute des commandes dans un sous-shell	(cd /tmp && ls)

$(...)	Substitution de commande	Remplace par la sortie de la commande	echo "Date: $(date)"
`...`	Substitution de commande (ancien)	Même chose que $(...) mais déprécié	echo "Date: `date` "
$VAR	Référence de variable	Remplace par la valeur de la variable	echo $HOME
${VAR}	Référence de variable	Version explicite pour éviter les ambiguïtés	echo ${HOME}texte
$?	Code de retour	Contient le code de retour de la dernière commande	echo $?
$$	PID du shell	Contient le PID du shell actuel	echo $$
$!	PID du dernier processus	Contient le PID du dernier processus lancé en arrière-plan	echo $!
$0	Nom du script	Contient le nom du script en cours d'exécution	echo $0
$1, $2, ...	Paramètres positionnels	Contiennent les arguments passés au script	echo $1
$#	Nombre de paramètres	Contient le nombre d'arguments passés au script	echo $#
$@	Tous les paramètres	Contient tous les arguments passés au script	echo $@
$*	Tous les paramètres	Similaire à $@ mais avec un traitement différent des guillemets	echo $*
#	Commentaire	Tout ce qui suit est ignoré (commentaire)	# Ceci est un commentaire
\	Échappement	Annule la signification spéciale du caractère suivant	echo *
'...'	Guillemets simples	Protège tous les caractères spéciaux (pas d'interprétation)	echo '$HOME'
"..."	Guillemets doubles	Protège la plupart des caractères mais permet les substitutions	echo "$HOME"
~	Tilde	Représente le répertoire personnel de l'utilisateur	cd ~
~user	Tilde + utilisateur	Représente le répertoire personnel de l'utilisateur spécifié	cd ~john
!	Substitution historique	Rappelle une commande de l'historique	!42 ou !ls
!!	Dernière commande	Rappelle la dernière commande exécutée	sudo !!
!$	Dernier argument	Rappelle le dernier argument de la commande précédente	ls -la /var/log/syslog; cat !$
^old^new	Substitution rapide	Répète la dernière commande en remplaçant "old" par "new"	^typo^correction

:	Commande nulle	Ne fait rien, retourne toujours vrai	: > fichier.txt (vide le fichier)
<<EOF	Here document	Entrée sur plusieurs lignes	cat <<EOF > fichier.txt
<<<	Here string	Fournit une chaîne comme entrée standard	grep "mot" <<< "Ceci contient un mot"
=	Affectation	Assigne une valeur à une variable	VAR=valeur
==	Égalité (test)	Compare si deux chaînes sont égales	if ["$VAR" == "valeur"]
-eq	Égalité numérique	Compare si deux nombres sont égaux	if ["$NUM" -eq 5]
-ne	Inégalité numérique	Compare si deux nombres sont différents	if ["$NUM" -ne 5]
-lt	Inférieur à	Compare si un nombre est inférieur à un autre	if ["$NUM" -lt 5]
-le	Inférieur ou égal	Compare si un nombre est inférieur ou égal à un autre	if ["$NUM" -le 5]
-gt	Supérieur à	Compare si un nombre est supérieur à un autre	if ["$NUM" -gt 5]
-ge	Supérieur ou égal	Compare si un nombre est supérieur ou égal à un autre	if ["$NUM" -ge 5]
-z	Chaîne vide	Teste si une chaîne est vide	if [-z "$VAR"]
-n	Chaîne non vide	Teste si une chaîne n'est pas vide	if [-n "$VAR"]
-e	Existence de fichier	Teste si un fichier existe	if [-e "$FILE"]
-f	Fichier ordinaire	Teste si le fichier existe et est un fichier ordinaire	if [-f "$FILE"]
-d	Répertoire	Teste si le fichier existe et est un répertoire	if [-d "$DIR"]
-r	Lisible	Teste si le fichier existe et est lisible	if [-r "$FILE"]
-w	Inscriptible	Teste si le fichier existe et est inscriptible	if [-w "$FILE"]
-x	Exécutable	Teste si le fichier existe et est exécutable	if [-x "$FILE"]

Exemples complexes de combinaisons:

Redirection de la sortie standard et d'erreur vers des fichiers différents
commande > sortie.txt 2> erreur.txt

Redirection de la sortie standard et d'erreur vers le même fichier
commande > fichier.txt 2>&1

```
# Pipe avec plusieurs commandes
cat fichier.txt | grep "motif" | sort | uniq -c

# Conditionnels avec commandes multiples
if [ -d "/tmp" ] && [ -w "/tmp" ]; then
    echo "Le répertoire /tmp existe et est inscriptible"
fi

# Substitution de commande complexe
echo "Il y a $(ls -la | wc -l) entrées dans le répertoire"

# Expansion de variables avec traitement
echo "${VAR:-valeur_defaut}"  # Utilise VAR s'il existe, sinon valeur_defaut
echo "${VAR:=valeur_defaut}"  # Assigne valeur_defaut à VAR s'il n'existe pas
echo "${VAR:+valeur_si_var_existe}"  # Utilise "valeur_si_var_existe" si VAR existe, sinon rien
echo "${VAR:0:5}"  # Extrait les 5 premiers caractères de VAR
```

Ce tableau couvre les opérateurs et caractères spéciaux les plus couramment utilisés dans Bash, avec des exemples pour illustrer leur utilisation.

Exemples détaillés pour ces opérateurs :

```
bash
# Pipe simple
ls -la | grep "\.txt$"

# Liste les fichiers et filtre pour ne garder que les .txt


# Pipe multiple
ps aux | grep apache | awk '{print $2}' | head -5

# Liste les processus, filtre Apache, extrait les PID, prend les 5 premiers


# OU logique - exécute la seconde commande seulement si la première échoue
mkdir /tmp/test || echo "Impossible de créer le répertoire"


# Combinaison OU et ET logique
cd /home/user && ls -la || echo "Impossible d'accéder au répertoire"
```