Liste des Variables BASH

Variables Bash

Ces variables sont définies ou utilisées par Bash, mais les autres shells ne les traitent normalement pas spécialement.

Quelques variables utilisées par Bash sont décrites dans différents chapitres : variables pour contrôler les fonctions de contrôle des tâches (voir Variables de contrôle des tâches).

_

($_, un trait de soulignement.) Au démarrage du shell, définit le chemin d'accès utilisé pour invoquer le shell ou le script shell en cours d'exécution tel que passé dans l'environnement ou la liste d'arguments. Par la suite, développe le dernier argument de la commande simple précédente exécutée au premier plan, après développement. Définit également le chemin d'accès complet utilisé pour invoquer chaque commande exécutée et placée dans l'environnement exporté vers cette commande. Lors de la vérification du courrier, ce paramètre contient le nom du fichier de courrier.

BASH

Le chemin d'accès complet utilisé pour exécuter l'instance actuelle de Bash.

BASHOPTS

Une liste séparée par deux points des options de shell activées. Chaque mot de la liste est un argument valide pour l'option **-s** de la commande intégrée shopt (voir La commande intégrée Shopt). Les options apparaissant dans **BASHOPTS** sont celles signalées comme « on » par « shopt ». Si cette variable est présente dans l'environnement au démarrage de Bash, chaque option de shell de la liste sera activée avant la lecture des fichiers de démarrage. Cette variable est en lecture seule.

BASHPID

S'étend à l'ID de processus du processus Bash actuel. Cela diffère de $$ dans certaines circonstances, comme les sous-shells qui ne nécessitent pas la réinitialisation de Bash. Les affectations à **BASHPID** n'ont aucun effet. Si **BASHPID** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_ALIASES

Une variable de tableau associatif dont les membres correspondent à la liste interne des alias telle que maintenue par la commande intégrée alias. (voir Bourne Shell Builtins). Les éléments ajoutés à ce

tableau apparaissent dans la liste des alias ; Cependant, la désactivation des éléments du tableau ne provoque pas actuellement la suppression des alias de la liste des alias. Si **BASH_ALIASES** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_ARGC

Une variable de tableau dont les valeurs sont le nombre de paramètres dans chaque image de la pile d'appels d'exécution bash actuelle. Le nombre de paramètres de la sous-routine actuelle (fonction shell ou script exécuté avec . ou source) est en haut de la pile. Lorsqu'une sous-routine est exécutée, le nombre de paramètres passés est poussé sur **BASH_ARGC**. Le shell définit **BASH_ARGC** uniquement en mode de débogage étendu (voir la fonction intégrée Shopt pour une description de l'option extdebug de la fonction intégrée shopt). La définition de extdebug après que le shell a commencé à exécuter un script, ou le référencement de cette variable lorsque extdebug n'est pas défini, peut entraîner des valeurs incohérentes.

BASH_ARGV

Une variable de tableau contenant tous les paramètres de la pile d'appels d'exécution bash actuelle. Le dernier paramètre du dernier appel de sous-routine se trouve en haut de la pile ; le premier paramètre de l'appel initial se trouve en bas. Lorsqu'une sous-routine est exécutée, les paramètres fournis sont placés sur **BASH_ARGV**. Le shell définit **BASH_ARGV** uniquement en mode débogage étendu (voir La fonction intégrée Shopt pour une description de l'option extdebug de la fonction intégrée shopt). La définition de extdebug après que le shell a commencé à exécuter un script, ou le référencement de cette variable lorsque extdebug n'est pas défini, peut entraîner des valeurs incohérentes.

BASH_ARGV0

Lorsqu'elle est référencée, cette variable se développe en nom du shell ou du script shell (identique à $0 ; voir Paramètres spéciaux, pour la description du paramètre spécial 0). L'affectation à **BASH_ARGV0** entraîne l'affectation de la valeur affectée à $0. Si **BASH_ARGV0** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_CMDS

Une variable de tableau associatif dont les membres correspondent à la table de hachage interne des commandes telle que maintenue par la fonction intégrée hash (voir Bourne Shell Builtins). Les éléments ajoutés à ce tableau apparaissent dans la table de hachage ; toutefois, la désactivation des éléments du tableau n'entraîne pas actuellement la suppression des noms de commande de la table de hachage. Si **BASH_CMDS** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_COMMAND

La commande en cours d'exécution ou sur le point d'être exécutée, sauf si le shell exécute une commande à la suite d'une interruption, auquel cas il s'agit de la commande exécutée au moment de l'interruption. Si **BASH_COMMAND** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_COMPAT

La valeur est utilisée pour définir le niveau de compatibilité du shell. Voir Mode de compatibilité du shell, pour une description des différents niveaux de compatibilité et de leurs effets. La valeur peut être un nombre décimal (par exemple, 4,2) ou un entier (par exemple, 42) correspondant au niveau de compatibilité souhaité. Si **BASH_COMPAT** n'est pas défini ou défini sur une chaîne vide, le niveau de compatibilité est défini sur la valeur par défaut de la version actuelle. Si **BASH_COMPAT** est défini sur une valeur qui n'est pas l'un des niveaux de compatibilité valides, le shell affiche un message d'erreur et définit le niveau de compatibilité sur la valeur par défaut de la version actuelle. r la version actuelle. Les valeurs valides correspondent aux niveaux de compatibilité décrits ci-dessous (voir Mode de compatibilité du shell). Par exemple, 4.2 et 42 sont des valeurs valides qui correspondent à l'option shopt compat42 et définissent le niveau de compatibilité à 42. La version actuelle est également une valeur valide.

BASH_ENV

Si cette variable est définie lorsque Bash est invoqué pour exécuter un script shell, sa valeur est développée et utilisée comme nom d'un fichier de démarrage à lire avant d'exécuter le script. Voir Fichiers de démarrage Bash.

BASH_EXECUTION_STRING

L'argument de commande de l'option d'invocation **-c**.

BASH_LINENO

Une variable de tableau dont les membres sont les numéros de ligne dans les fichiers source où chaque membre correspondant de **FUNCNAME** a été invoqué. `${BASH_LINENO[$i]}` est le numéro de ligne dans le fichier source (`${BASH_SOURCE[$i+1]}`) où `${FUNCNAME[$i]}` a été appelé (ou `${BASH_LINENO[$i-1]}` s'il est référencé dans une autre fonction shell). Utiliser LINENO pour obtenir le numéro de ligne actuel.

BASH_LOADABLES_PATH

Une liste séparée par deux points de répertoires dans lesquels le shell recherche les fonctions intégrées chargeables dynamiquement spécifiées par la commande enable.

BASH_REMATCH

Une variable de tableau dont les membres sont assignés par l'opérateur binaire ‘=~’ à la commande conditionnelle [[(voir Constructions conditionnelles). L'élément avec l'index 0 est la partie de la chaîne correspondant à l'expression régulière entière. L'élément avec l'index n est la partie de la chaîne correspondant à la n-ième sous-expression entre parenthèses.

BASH_SOURCE

Une variable de tableau dont les membres sont les noms de fichiers sources où les noms de fonctions shell correspondants dans la variable de tableau FUNCNAME sont définis. La fonction shell `${FUNCNAME[$i]}` est définie dans le fichier `${BASH_SOURCE[$i]}` et appelée depuis `${BASH_SOURCE[$i+1]}`

```
#!/bin/bash -Eeux
echo ${BASH_SOURCE}
echo $0
```

retourne

```
+ echo ./test
./test
+ echo ./test
./test
```

BASH_SUBSHELL

Incrémentée de un dans chaque sous-shell ou environnement de sous-shell lorsque le shell commence à s'exécuter dans cet environnement. La valeur initiale est 0. Si BASH_SUBSHELL n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

BASH_VERSINFO

Une variable de tableau en lecture seule (voir Tableaux) dont les membres contiennent des informations de version pour cette instance de Bash. Les valeurs attribuées aux membres du tableau sont les suivantes :

BASH_VERSINFO[0]

Le numéro de version majeur (la version).

BASH_VERSINFO[1]

Le numéro de version mineur (la version).

BASH_VERSINFO[2]

Le niveau de correctif.

BASH_VERSINFO[3]

La version de build.

BASH_VERSINFO[4]

L'état de publication (par exemple, beta1).

BASH_VERSINFO[5]

La valeur de MACHTYPE.

BASH_VERSION

Le numéro de version de l'instance actuelle de Bash.

BASH_XTRACEFD

Si défini sur un entier correspondant à un descripteur de fichier valide, Bash écrira la sortie de trace générée lorsque « set -x » est activé sur ce descripteur de fichier. Cela permet de séparer la sortie de trace des messages de diagnostic et d'erreur. Le descripteur de fichier est fermé lorsque **BASH_XTRACEFD** est désactivé ou qu'une nouvelle valeur lui est attribuée. La désactivation de **BASH_XTRACEFD** ou l'affectation d'une chaîne vide entraîne l'envoi de la sortie de trace à l'erreur standard.

La définition de **BASH_XTRACEFD** sur 2 (le descripteur de fichier d'erreur standard) puis sa désactivation entraînera la fermeture de l'erreur standard.

CHILD_MAX

Définit le nombre de valeurs d'état enfant quittées que le shell doit mémoriser. Bash ne permettra pas que cette valeur soit diminuée en dessous d'un minimum imposé par POSIX, et il existe une valeur maximale (actuellement 8192) que cette valeur ne peut pas dépasser. La valeur minimale dépend du système.

COLUMNS

Utilisé par la commande select pour déterminer la largeur du terminal lors de l'impression des listes de sélection. Défini automatiquement si l'option checkwinsize est activée (voir The Shopt Builtin), ou dans un shell interactif à la réception d'un **SIGWINCH**.

COMP_CWORD

Un index dans `${COMP_WORDS}` du mot contenant la position actuelle du curseur. Cette variable est disponible uniquement dans les fonctions shell invoquées par les fonctions de complétion programmables (voir Complétion programmable).

COMP_LINE

La ligne de commande actuelle. Cette variable est disponible uniquement dans les fonctions shell et les commandes externes invoquées par les fonctions de complétion programmables (voir Complétion programmable).

COMP_POINT

L'index de la position actuelle du curseur par rapport au début de la commande en cours. Si la position actuelle du curseur se trouve à la fin de la commande en cours, la valeur de cette variable est égale à `${#COMP_LINE}`. Cette variable est disponible uniquement dans les fonctions shell et les commandes externes invoquées par les fonctions de complétion programmables (voir Complétion programmable).

COMP_TYPE

Définit une valeur entière correspondant au type de complétion tentée qui a provoqué l'appel d'une fonction de complétion : TAB, pour la complétion normale, « ? », pour lister les complétions après des tabulations successives, « ! », pour lister les alternatives sur la complétion partielle d'un mot, « @ », pour lister les complétions si le mot n'est pas non modifié, ou « % », pour la complétion de menu. Cette variable est disponible uniquement dans les fonctions shell et les commandes externes invoquées par les fonctions de complétion programmables (voir Pr

COMP_KEY

La touche (ou la touche finale d'une séquence de touches) utilisée pour invoquer la fonction de

complétion en cours.

COMP_WORDBREAKS

L'ensemble de caractères que la bibliothèque Readline traite comme séparateurs de mots lors de l'exécution de la complétion de mots. Si **COMP_WORDBREAKS** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

COMP_WORDS

Une variable de tableau composée des mots individuels de la ligne de commande en cours. La ligne est divisée en mots comme Readline la diviserait, en utilisant **COMP_WORDBREAKS** comme décrit ci-dessus. Cette variable n'est disponible que dans les fonctions shell invoquées par les fonctions de complétion programmables (voir Complétion programmable).

COMPREPLY

Une variable de tableau à partir de laquelle Bash lit les complétions possibles générées par une fonction shell invoquée par la fonction de complétion programmable (voir Complétion programmable). Chaque élément de tableau contient une complétion possible.

COPROC

Une variable de tableau créée pour contenir les descripteurs de fichiers pour la sortie et l'entrée dans un coprocessus sans nom (voir Coprocessus).

DIRSTACK

Une variable de tableau contenant le contenu actuel de la pile de répertoires. Les répertoires apparaissent dans la pile dans l'ordre dans lequel ils sont affichés par la fonction intégrée dirs. L'affectation aux membres de cette variable de tableau peut être utilisée pour modifier les répertoires déjà présents dans la pile, mais les fonctions intégrées pushd et popd doivent être utilisées pour ajouter et supprimer des répertoires. L'affectation à cette variable ne modifiera pas le répertoire actuel. Si DIRSTACK n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

EMACS

Si Bash trouve cette variable dans l'environnement lorsque le shell démarre avec la valeur « t », il suppose que le shell s'exécute dans un tampon de shell Emacs et désactive l'édition de ligne.

ENV

Étendu et exécuté de manière similaire à **BASH_ENV** (voir Fichiers de démarrage de Bash) lorsqu'un shell interactif est invoqué en mode POSIX (voir Mode POSIX de Bash).

EPOCHREALTIME

Chaque fois que ce paramètre est référencé, il s'étend au nombre de secondes depuis l'époque Unix sous forme de valeur à virgule flottante avec une granularité de l'ordre de la micro-seconde. Les affectations à EPOCHREALTIME sont ignorées. Si **EPOCHREALTIME** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

EPOCHSECONDS

Chaque fois que ce paramètre est référencé, il s'étend au nombre de secondes depuis l'époque Unix. Les affectations à **EPOCHSECONDS** sont ignorées. Si **EPOCHSECONDS** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

EUID

L'ID utilisateur effectif numérique de l'utilisateur actuel. Cette variable est en lecture seule.

EXECIGNORE

Liste séparée par deux points de modèles de shell (voir Correspondance de modèles) définissant la liste des noms de fichiers à ignorer par la recherche de commande à l'aide de PATH. Les fichiers dont les noms de chemin complets correspondent à l'un de ces modèles ne sont pas considérés comme des fichiers exécutables aux fins de la complétion et de l'exécution de commandes via la recherche PATH. Cela n'affecte pas le comportement des commandes [, test et [[. Les noms de chemin complets dans la table de hachage de commande ne sont pas soumis à EXECIGNORE. Utiliser cette variable pour ignorer les fichiers de bibliothèque partagée dont le bit exécutable est défini, mais qui ne sont pas des fichiers exécutables. La correspondance de modèles respecte le paramètre de l'option shell extglob.

FCEDIT

L'éditeur utilisé par défaut par l'option -e de la commande intégrée fc.

FIGNORE

Liste séparée par deux points de suffixes à ignorer lors de la complétion de nom de fichier. Un nom de fichier dont le suffixe correspond à l'une des entrées de FIGNORE est exclu de la liste des noms de

fichiers correspondants. Un exemple de valeur est « .o:~ »

FUNCNAME

Une variable de tableau contenant les noms de toutes les fonctions shell actuellement dans la pile d'appels d'exécution. L'élément avec l'index 0 est le nom de toute fonction shell en cours d'exécution. L'élément le plus bas (celui avec l'index le plus élevé) est « main ». Cette variable n'existe que lorsqu'une fonction shell est en cours d'exécution. Les affectations à FUNCNAME n'ont aucun effet. Si FUNCNAME n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

Cette variable peut être utilisée avec **BASH_LINENO** et **BASH_SOURCE**. Chaque élément de **FUNCNAME** a des éléments correspondants dans **BASH_LINENO** et **BASH_SOURCE** pour décrire la pile d'appels. Par exemple, `${FUNCNAME[$i]}` a été appelé à partir du fichier `${BASH_SOURCE[$i+1]}` à la ligne numéro `${BASH_LINENO[$i]}`. La fonction intégrée caller affiche la pile d'appels actuelle à l'aide de ces informations.

FUNCNEST

Si la valeur est supérieure à 0, définit un niveau d'imbrication de fonction maximal. Les appels de fonction qui dépassent ce niveau d'imbrication entraîneront l'abandon de la commande en cours.

GLOBIGNORE

Une liste de modèles séparés par deux points définissant l'ensemble des noms de fichiers à ignorer par l'expansion du nom de fichier. Si un nom de fichier correspondant à un modèle d'expansion de nom de fichier correspond également à l'un des modèles de **GLOBIGNORE**, il est supprimé de la liste des correspondances. La correspondance de modèle respecte le paramètre de l'option de shell **extglob**.

GROUPS

Une variable de tableau contenant la liste des groupes dont l'utilisateur actuel est membre. Les affectations à GROUPS n'ont aucun effet. Si GROUPS n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

histchars

Jusqu'à trois caractères qui contrôlent l'expansion de l'historique, la substitution rapide et la tokenisation (voir Expansion de l'historique). Le premier caractère est le caractère d'expansion de l'historique, c'est-à-dire le caractère qui signifie le début d'une expansion de l'historique, normalement « ! ». Le deuxième caractère est le caractère qui signifie « substitution rapide » lorsqu'il est vu comme le premier caractère d'une ligne, normalement « ^ ». Le troisième caractère facultatif est le caractère qui indique que le reste de la ligne est un commentaire lorsqu'il est trouvé comme le premier caractère d'un mot, généralement « # ». Le caractère de commentaire de l'historique fait que

la substitution de l'historique est ignorée pour les mots restants de la ligne. Il n'oblige pas nécessairement l'analyseur du shell à traiter le reste de la ligne comme un commentaire.

HISTCMD

Le numéro d'historique, ou l'index dans la liste d'historique, de la commande en cours. Les affectations à **HISTCMD** sont ignorées. Si **HISTCMD** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

HISTCONTROL

Une liste de valeurs séparées par deux points qui contrôle la manière dont les commandes sont enregistrées dans la liste d'historique. Si la liste de valeurs inclut « ignorespace », les lignes qui commencent par un espace ne sont pas enregistrées dans la liste d'historique. Une valeur de « ignoredups » entraîne la non-enregistrement des lignes qui correspondent à l'entrée d'historique précédente. Une valeur de « ignoreboth » est un raccourci pour « ignorespace » et « ignoredups ». Une valeur de « erasedups » entraîne la suppression de toutes les lignes précédentes correspondant à la ligne actuelle de la liste d'historique avant l'enregistrement de cette ligne. Toute valeur qui ne figure pas dans la liste ci-dessus est ignorée. Si **HISTCONTROL** n'est pas défini ou n'inclut pas de valeur valide, toutes les lignes lues par l'analyseur de shell sont enregistrées dans la liste d'historique, sous réserve de la valeur de **HISTIGNORE**. La deuxième ligne et les suivantes d'une commande composée de plusieurs lignes ne sont pas testées et sont ajoutées à l'historique quelle que soit la valeur de **HISTCONTROL**.

HISTFILE

Le nom du fichier dans lequel l'historique des commandes est enregistré. La valeur par défaut est `~/.bash_history`.

HISTFILESIZE

Le nombre maximal de lignes contenues dans le fichier d'historique. Lorsque cette variable reçoit une valeur, le fichier d'historique est tronqué, si nécessaire, pour ne pas contenir plus que ce nombre de lignes en supprimant les entrées les plus anciennes. Le fichier d'historique est également tronqué à cette taille après l'avoir écrit lorsqu'un shell se ferme. Si la valeur est 0, le fichier d'historique est tronqué à une taille zéro. Les valeurs non numériques et les valeurs numériques inférieures à zéro empêchent la troncation. Le shell définit la valeur par défaut sur la valeur de **HISTSIZE** après avoir lu les fichiers de démarrage.

HISTIGNORE

Une liste de modèles séparés par deux points utilisée pour décider quelles lignes de commande doivent être enregistrées dans la liste d'historique. Chaque modèle est ancré au début de la ligne et doit correspondre à la ligne complète (aucun « * » implicite n'est ajouté). Chaque modèle est testé

par rapport à la ligne après l'application des vérifications spécifiées par **HISTCONTROL**. En plus des caractères de correspondance de modèle de shell normaux, « & » correspond à la ligne d'historique précédente. « & » peut être échappé à l'aide d'une barre oblique inverse ; la barre oblique inverse est supprimée avant de tenter une correspondance. La deuxième ligne et les suivantes d'une commande composée de plusieurs lignes ne sont pas testées et sont ajoutées à l'historique quelle que soit la valeur de **HISTIGNORE**. La correspondance de modèle respecte le paramètre de l'option shell extglob.

HISTIGNORE englobe la fonction de **HISTCONTROL**. Un modèle de « & » est identique à ignoredups, et un modèle de « [ ]* » est identique à ignorespace. La combinaison de ces deux modèles, en les séparant par deux points, fournit la fonctionnalité d'ignoreboth.

HISTSIZE

Le nombre maximal de commandes à mémoriser dans la liste d'historique. Si la valeur est 0, les commandes ne sont pas enregistrées dans la liste d'historique. Les valeurs numériques inférieures à zéro entraînent l'enregistrement de toutes les commandes dans la liste d'historique (il n'y a pas de limite). Le shell définit la valeur par défaut à 500 après avoir lu tous les fichiers de démarrage.

HISTTIMEFORMAT

Si cette variable est définie et non nulle, sa valeur est utilisée comme chaîne de format pour strftime pour imprimer l'horodatage associé à chaque entrée d'historique affichée par la fonction intégrée history. Si cette variable est définie, les horodatages sont écrits dans le fichier d'historique afin qu'ils puissent être conservés entre les sessions du shell. Cela utilise le caractère de commentaire d'historique pour distinguer les horodatages des autres lignes d'historique.

HOSTFILE

Contient le nom d'un fichier au même format que /etc/hosts qui doit être lu lorsque le shell doit compléter un nom d'hôte. La liste des complétions de noms d'hôtes possibles peut être modifiée pendant que le shell est en cours d'exécution ; la prochaine fois que la complétion de nom d'hôte est tentée après que la valeur a été modifiée, Bash ajoute le contenu du nouveau fichier à la liste existante. Si HOSTFILE est défini, mais n'a aucune valeur ou ne nomme pas de fichier lisible, Bash tente de lire /etc/hosts pour obtenir la liste des noms d'hôtes possibles achèvements. Lorsque HOSTFILE n'est pas défini, la liste des noms d'hôtes est effacée.

HOSTNAME

Le nom de l'hôte actuel.

HOSTTYPE

Une chaîne décrivant la machine sur laquelle Bash s'exécute.

IGNOREEOF

Contrôle l'action du shell à la réception d'un caractère EOF comme seule entrée. Si elle est définie, la valeur indique le nombre de caractères EOF consécutifs qui peuvent être lus comme premier caractère sur une ligne d'entrée avant que le shell ne se ferme. Si la variable existe mais n'a pas de valeur numérique, ou n'a pas de valeur, alors la valeur par défaut est 10. Si la variable n'existe pas, alors EOF signifie la fin de l'entrée dans le shell. Ceci n'est en vigueur que pour les shells interactifs.

INPUTRC

Le nom du fichier d'initialisation Readline, remplaçant la valeur par défaut de `~/.inputrc`.

INSIDE_EMACS

Si Bash trouve cette variable dans l'environnement au démarrage du shell, il suppose que le shell s'exécute dans un tampon shell Emacs et peut désactiver l'édition de ligne en fonction de la valeur de **TERM**.

LANG

Utilisé pour déterminer la catégorie de paramètres régionaux pour toute catégorie non spécifiquement sélectionnée avec une variable commençant par LC_.

LC_ALL

Cette variable remplace la valeur de LANG et toute autre variable LC_ spécifiant une catégorie de paramètres régionaux.

LC_COLLATE

Cette variable détermine l'ordre de classement utilisé lors du tri des résultats de l'expansion du nom de fichier, et détermine le comportement des expressions de plage, des classes d'équivalence et des séquences de classement dans l'expansion du nom de fichier et la correspondance de motifs (voir Expansion du nom de fichier).

LC_CTYPE

Cette variable détermine l'interprétation des caractères et le comportement des classes de caractères dans l'expansion du nom de fichier et la correspondance de motifs (voir Expansion du nom de fichier).

LC_MESSAGES

Cette variable détermine les paramètres régionaux utilisés pour traduire les chaînes entre guillemets précédées d'un « $ » (voir Traduction spécifique aux paramètres régionaux).

LC_NUMERIC

Cette variable détermine la catégorie de paramètres régionaux utilisée pour le formatage des nombres.

LC_TIME

Cette variable détermine la catégorie de paramètres régionaux utilisée pour le formatage des données et de l'heure.

LINENO

Le numéro de ligne dans le script ou la fonction shell en cours d'exécution. Si LINENO n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

LINES

Utilisé par la commande select pour déterminer la longueur de colonne pour l'impression des listes de sélection. Défini automatiquement si l'option checkwinsize est activée (voir The Shopt Builtin), ou dans un shell interactif à la réception d'un SIGWINCH.

MACHTYPE

Une chaîne qui décrit entièrement le type de système sur lequel Bash s'exécute, au format standard GNU cpu-company-system.

MAILCHECK

Fréquence (en secondes) à laquelle le shell doit vérifier le courrier dans les fichiers spécifiés dans les variables MAILPATH ou MAIL. La valeur par défaut est de 60 secondes. Lorsqu'il est temps de vérifier le courrier, le shell le fait avant d'afficher l'invite principale. Si cette variable n'est pas définie ou définie sur une valeur qui n'est pas un nombre supérieur ou égal à zéro, le shell désactive la vérification du courrier.

MAPFILE

Une variable de tableau créée pour contenir le texte lu par la commande intégrée mapfile lorsqu'aucun nom de variable n'est fourni.

OLDPWD

Le répertoire de travail précédent tel que défini par la commande intégrée cd.

OPTERR

Si elle est définie sur la valeur 1, Bash affiche les messages d'erreur générés par la commande intégrée getopts.

OSTYPE

Une chaîne décrivant le système d'exploitation sur lequel Bash s'exécute.

PIPESTATUS

Une variable de tableau (voir Tableaux) contenant une liste de valeurs d'état de sortie des processus du pipeline de premier plan le plus récemment exécuté (qui ne peut contenir qu'une seule commande).

POSIXLY_CORRECT

Si cette variable est dans l'environnement au démarrage de Bash, le shell entre en mode POSIX (voir Mode POSIX de Bash) avant de lire les fichiers de démarrage, comme si l'option d'invocation –posix avait été fournie. Si elle est définie pendant que le shell est en cours d'exécution, Bash active le mode POSIX, comme si la commande

```
set -o posix
```

avait été exécutée. Lorsque le shell entre en mode POSIX, il définit cette variable si elle n'était pas déjà définie.

PPID

L'ID de processus du processus parent du shell. Cette variable est en lecture seule.

PROMPT_COMMAND

Si cette variable est définie et qu'elle est un tableau, la valeur de chaque élément défini est interprétée comme une commande à exécuter avant d'imprimer l'invite principale (`$PS1`). Si elle est définie mais n'est pas une variable de tableau, sa valeur est utilisée comme une commande à exécuter à la place.

PROMPT_DIRTRIM

Si elle est définie sur un nombre supérieur à zéro, la valeur est utilisée comme le nombre de composants de répertoire de fin à conserver lors de l'expansion des échappements de chaîne d'invite \w et \W (voir Contrôle de l'invite). Les caractères supprimés sont remplacés par des points de suspension.

PS0

La valeur de ce paramètre est développée comme PS1 et affichée par les shells interactifs après la lecture d'une commande et avant l'exécution de la commande.

PS3

La valeur de cette variable est utilisée comme invite pour la commande select. Si cette variable n'est pas définie, la commande select affiche une invite avec « # ? »

PS4

La valeur de ce paramètre est développée comme PS1 et la valeur développée est l'invite imprimée avant que la ligne de commande ne soit affichée lorsque l'option -x est définie (voir Le paramètre Set Builtin). Le premier caractère de la valeur étendue est répliqué plusieurs fois, si nécessaire, pour indiquer plusieurs niveaux d'indirection. La valeur par défaut est « + ».

PWD

Le répertoire de travail actuel tel que défini par la fonction intégrée cd.

RANDOM

Chaque fois que ce paramètre est référencé, il se développe en un entier aléatoire compris entre 0 et 32767. L'attribution d'une valeur à cette variable amorce le générateur de nombres aléatoires. Si **RANDOM** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

READLINE_ARGUMENT

Tout argument numérique donné à une commande Readline qui a été définie à l'aide de « bind -x » (voir Commandes intégrées de Bash lors de son appel.

READLINE_LINE

Le contenu du tampon de ligne Readline, à utiliser avec « bind -x » (voir Commandes intégrées de Bash).

READLINE_MARK

La position de la marque (point d'insertion enregistré) dans le tampon de ligne Readline, à utiliser avec « bind -x » (voir Commandes intégrées de Bash). Les caractères entre le point d'insertion et la marque sont souvent appelés la région.

READLINE_POINT

La position du point d'insertion dans le tampon de ligne Readline, à utiliser avec « bind -x » (voir Commandes intégrées de Bash).

REPLY

La variable par défaut pour la commande intégrée read.

SECONDS

Cette variable s'étend au nombre de secondes depuis le démarrage du shell. L'affectation à cette variable réinitialise le compteur à la valeur attribuée, et la valeur étendue devient la valeur attribuée plus le nombre de secondes depuis l'affectation. Le nombre de secondes à l'appel du shell et l'heure actuelle sont toujours déterminés en interrogeant l'horloge système. Si **SECONDS** n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

SHELL

Cette variable d'environnement s'étend au chemin d'accès complet au shell. Si elle n'est pas définie au démarrage du shell, Bash lui attribue le chemin d'accès complet du shell de connexion de l'utilisateur actuel.

SHELLOPTS

Une liste séparée par deux points d'options de shell activées. Chaque mot de la liste est un argument valide pour l'option -o de la commande intégrée set (voir La commande intégrée Set). Les options

apparaissant dans SHELLOPTS sont celles signalées comme « on » par « set -o ». Si cette variable est dans l'environnement au démarrage de Bash, chaque option de shell de la liste sera activée avant de lire les fichiers de démarrage. Cette variable est en lecture seule. SHLVL

Incrémentée d'une unité à chaque démarrage d'une nouvelle instance de Bash. Il s'agit d'un décompte de la profondeur d'imbrication de vos shells Bash.

SRANDOM

Cette variable se développe en un nombre pseudo-aléatoire de 32 bits à chaque fois qu'elle est référencée. Le générateur de nombres aléatoires n'est pas linéaire sur les systèmes qui prennent en charge /dev/urandom ou arc4random, donc chaque nombre renvoyé n'a aucune relation avec les nombres qui le précèdent. Le générateur de nombres aléatoires ne peut pas être initialisé, donc les affectations à cette variable n'ont aucun effet. Si SRANDOM n'est pas défini, il perd ses propriétés spéciales, même s'il est réinitialisé par la suite.

TIMEFORMAT

La valeur de ce paramètre est utilisée comme chaîne de format spécifiant comment les informations de temporisation pour les pipelines préfixés par le mot réservé time doivent être affichées. Le caractère « % » introduit une séquence d'échappement qui est développée en une valeur de temps ou d'autres informations. Les séquences d'échappement et leurs significations sont les suivantes ; les accolades désignent les parties facultatives.

%%	Un «% » littéral.
%[p][l]R	Le temps écoulé en secondes.
%[p][l]U	Le nombre de secondes CPU passées en mode utilisateur.
%[p][l]S	Le nombre de secondes CPU passées en mode système.
%P	Le pourcentage CPU, calculé comme (%U + %S) / %R.

Le **p** facultatif est un chiffre spécifiant la précision, le nombre de chiffres fractionnaires après un point décimal. Une valeur de 0 n'entraîne pas l'affichage d'un point décimal ou d'une fraction. Au plus trois positions après le point décimal peuvent être spécifiées ; les valeurs de **p** supérieures à 3 sont modifiées en 3. Si **p** n'est pas spécifié, la valeur 3 est utilisée.

Le **l** facultatif spécifie un format plus long, incluant les minutes, de la forme **MMmSS.FFs**. La valeur de **p** détermine si la fraction est incluse ou non.

Si cette variable n'est pas définie, Bash agit comme si elle avait la valeur

```
$'\nreal\t%3lR\nuser\t%3lU\nsys\t%3lS'
```

Si la valeur est nulle, aucune information de temporisation n'est affichée. Une nouvelle ligne de fin est ajoutée lorsque la chaîne de format est affichée.

TMOUT

Si la valeur est supérieure à zéro, TMOUT est considéré comme le délai d'attente par défaut pour la commande intégrée read (voir Commandes intégrées Bash). La commande select (voir Constructions conditionnelles) se termine si l'entrée n'arrive pas après **TMOUT** secondes lorsque l'entrée provient d'un terminal.

Dans un shell interactif, la valeur est interprétée comme le nombre de secondes à attendre pour une ligne d'entrée après l'émission de l'invite principale. Bash se termine après avoir attendu ce nombre de secondes si une ligne d'entrée complète n'arrive pas.

TMPDIR

Si elle est définie, Bash utilise sa valeur comme nom d'un répertoire dans lequel Bash crée des fichiers temporaires pour l'utilisation du shell.

UID

L'ID utilisateur réel numérique de l'utilisateur actuel. Cette variable est en lecture seule.

Récapitulatif

variables	Détails
$* / $@	Paramètres positionnels de fonction/script (arguments) : $* et $@ sont identiques à $1 $2 … (cela n'a généralement aucun sens de les laisser sans guillemets) “$*” est identique à “$1 $2 …” (Les arguments sont séparés par le premier caractère de $IFS, qui ne doit pas nécessairement être un espace) “@” est identique à “$1” “$2” …
$#	Nombre de paramètres positionnels passés au script ou à la fonction
$!	ID de processus de la dernière commande (la plus à droite pour les pipelines) dans le travail le plus récemment mis en arrière-plan (notez qu'il n'est pas nécessairement le même que l'ID de groupe de processus du travail lorsque le contrôle des travaux est activé)
$$	ID du processus qui a exécuté bash
$?	État de sortie de la dernière commande
$n	Paramètres positionnels, où n=1, 2, 3, …, 9
${n}	Paramètres positionnels (comme ci-dessus), mais n peut être > 9
$0	Dans les scripts, chemin avec lequel le script a été appelé ; avec `bash -c 'printf "%s\n" "$0"' name args`: name (le premier argument après le script en ligne), sinon, l'argv[0] que bash a reçu.
$_	Dernier champ de la dernière commande
$IFS	Séparateur de champ interne
$PATH	Variable d'environnement PATH utilisée pour rechercher des exécutables
$OLDPWD	Répertoire de travail précédent
$PWD	Présente le répertoire de travail

variables	Détails
$FUNCNAME	Tableau de noms de fonctions dans la pile des appels d'exécution
$BASH_SOURCE	Tableau contenant les chemins source des éléments du tableau FUNCNAME. Peut être utilisé pour obtenir le chemin du script. [1)]
$BASH_ALIASES	Tableau associatif contenant tous les alias actuellement définis
$BASH_REMATCH	Tableau des correspondances de la dernière correspondance de regex
$BASH_VERSION	Chaîne de version bash
$BASH_VERSINFO	Un tableau de 6 éléments avec les informations de version de Bash
$BASH	Chemin absolu vers le shell Bash en cours d'exécution lui-même (déterminé de manière heuristique par bash basé sur argv[0] et la valeur de $PATH ; peut être erroné dans les cas extrêmes)
$BASH_SUBSHELL	Niveau du sous-shell bash
$UID	Réel (inefficace si différent) ID utilisateur du processus exécutant bash
$PS1	Invite de ligne de commande principale ; voir Utilisation des variables PS*
$PS2	Invite de ligne de commande secondaire (utilisée pour une entrée supplémentaire)
$PS3	Invite de ligne de commande tertiaire (utilisée dans la boucle de sélection)
$PS4	Invite de ligne de commande Quaternaire (utilisée pour ajouter des informations avec une sortie détaillée)
$RANDOM	Un entier pseudo-aléatoire entre 0 et 32767
$REPLY	Variable utilisée par lecture par défaut lorsqu'aucune variable n'est spécifiée. Également utilisé par select pour renvoyer la valeur fournie par l'utilisateur
$PIPESTATUS	Variable de tableau contenant les valeurs d'état de sortie de chaque commande dans le pipeline de premier plan le plus récemment exécuté.

L'affectation de variable ne doit pas avoir d'espace avant et après. a = 123 et non a = 123. Ce dernier (un signe égal entouré d'espaces) signifie isolément exécuter la commande a avec les arguments = et 123, bien qu'il soit également vu dans l'opérateur de comparaison de chaînes (qui syntaxiquement est un argument de [ou [[ou selon le test utiliseé).

[1)]

$BASH_SOURCE contient comme *$0**le path d'exécution mais fait partie de la spécification du shell POSIX, alors que $BASH_SOURCE, comme son nom l'indique, est spécifique à Bash. De plus, *$0 peut être défini sur une valeur arbitraire par l'appelant. D'un autre côté, $BASH_SOURCE peut être vide, si aucun fichier nommé n'est impliqué; par exemple: `echo 'echo "[$BASH_SOURCE]"' | bash`