Les variables d'environnement

DEBIAN : Apprendre à rédiger des scripts sous bash FR

DEBIAN :Variables environnemnt FR

Les variables d'environnement constituent un moyen d'influencer le comportement des logiciels sur votre système. Par exemple, la variable d'environnement « LANG » détermine la langue que les logiciels utilisent pour communiquer avec l'utilisateur.

Les variables sont constituées de **noms** auxquels on assigne des **valeurs**. Ainsi, le système d'un utilisateur français devrait avoir la valeur « fr_FR.UTF-8 » assignée à la variable « LANG ».

La signification d'une variable d'environnement et le type de valeur qui peut lui être assignée sont déterminés par l'application qui utilise celle-ci. Il existe un petit nombre de variables d'environnement bien connues, dont le sens et le type de valeur sont bien déterminés, et qui sont utilisées par de nombreuses applications.

Manipuler les variables d'environnement

Bien que quelques applications de configuration en mode graphique manipulent en réalité des variables d'environnement, la ligne de commande offre un maximum de flexibilité pour créer et modifier ces variables.

Les techniques expliquées ci-dessous s'appliquent à la famille d'interpréteurs de commandes « Bourne Shell », c'est à dire **sh**, **ksh**, et **bash**. Ce dernier est l'interpréteur de commande par défaut d'Ubuntu. Si vous utilisez un autre interpréteur de commande, comme **csh**, les commandes indiquées pourraient être différentes.

Assigner des valeurs aux variables d'environnement

Pour affecter une valeur à une variable d'environnement **existante**, nous utilisons une expression d'affectation. Par exemple pour assigner la valeur « fr_FR.UTF-8 » à la variable « LANG », nous utilisons la commande suivante :

```
LANG=fr_FR.UTF-8
```

Si nous utilisons une expression d'affectation pour une variable qui n'existe pas, l'interpréteur de commande créera une « variable de shell », qui est similaire à une variable d'environnement mais qui n'affectera pas le comportement d'autres applications.

Une variable de shell peut être exportée pour devenir une variable d'environnement grâce à la commande **export**. Pour créer la variable d'environnement « EDITOR » et lui assigner la valeur « nano », plusieurs méthodes peuvent être utilisée. Voici celle que nous proposons :

```
EDITOR=nano
export EDITOR
```

L'interpréteur de commandes **bash** (celui qui est fourni par défaut avec Ubuntu) propose un raccourci pour créer des variables d'environnement. L'exemple précédent peut être condensé en une seule ligne de commande :

```
export EDITOR=nano
```

Examiner les valeurs des variables d'environnement

La commande **printenv** affiche les noms et les valeurs de toutes les variables d'environnement définies :

```
printenv
```

Pour examiner la valeur d'une variable en particulier, il suffit de spécifier son nom après la commande **printenv** :

```
printenv TERM
```

La valeur de la variable peut également être récupérée en utilisant le signe « $ » devant son nom, comme dans l'exemple suivant :

```
echo $TERM
```

Il existe une commande qui permet de faire des modifications temporaires, à court terme, sur l'environnement. Cela peut aussi être utilisé pour afficher les variables d'environnement courantes. Il s'agit de la commande **env** :

```
env
```

Le signe dollar peut être utilisé pour récupérer la valeur des variables d'environnement dans les lignes de commandes. Par exemple, la commande suivante peut être utilisée pour lister les fichiers du bureau (dossier Desktop) de l'utilisateur courant :

```
ls $HOME/Desktop
```

Effacer des variables d'environnement

Dans la plupart des cas, affecter une valeur vide à une variable d'environnement suffit à annuler son effet, comme dans l'exemple ci-dessous. Cependant certaines variables comme « POSIXLY_CORRECT » ont une influence sur les programmes du fait même de leur existence, et ce, même si leur valeur est vide.

```
export LC_ALL=
```

La commande **unset** peut être utilisée pour supprimer complètement une variable d'environnement :

```
unset LCALL
```

Il est également possible d'utiliser la commande **export** avec l'argument **-n**. Ceci aura pour effet de faire perdre à la variable son statut de variable d'environnement, elle devient une variable de shell tout en conservant sa valeur.

```
export -n LC_ALL
```

Principe de fonctionnement des variables d'environnement

Le fonctionnement et l'effet des variables d'environnement sont gouvernés par quelques principes simples.

Portée des variables

Les variables d'environnement ont une portée locale. Ce qui signifie que leur valeur est spécifique au processus dans lequel ou pour lequel elles ont été définies. Ainsi si vous ouvrez deux terminaux différents, c'est à dire deux processus **bash** différents, et que vous changez la valeur d'une variable d'environnement dans un terminal, ce changement n'affectera pas l'autre terminal ni aucun autre programme. Ce changement est local, il affecte le processus dans lequel il a été effectué, sans aucune influence sur les autres processus externes.

Héritage

Lorsqu'un processus enfant est créé à partir d'un processus parent, le processus enfant hérite de toutes les variables du processus parent, avec leurs valeurs. Par exemple, si on lance **gedit** depuis un terminal, **bash** le processus parent, engendre le processus enfant **gedit**.

En conséquence, si nous définissons la valeur de la variable d'environnement « LANG » dans un terminal, et que nous lançons depuis le même terminal **gedit**, celui-ci héritera de la nouvelle valeur de la variable LANG, et s'affichera donc dans une langue différente du reste des applications du système.

Notez bien que du fait de la portée des variables, expliquée plus haut, une fois le processus **gedit** lancé, les modifications de variables dans le processus parent ne seront pas répercutées sur le processus enfant et vice-versa.

Dans l'environnement de bureau GNOME, « gnome-session » est le processus parent de tous les processus s'éxecutant dans cet environnement. Ceci constitue (avec le principe d'héritage) le point clé qui nous permet d'influencer le comportement de notre environnement de bureau grâce aux variables d'environnement. L'équivalent pour KDE est « kde-session ».

Sensibilité à la casse

Le nom des variables d'environnement est sensible à la casse. Ce qui signifie que *LANG* n'est pas la même variable que *lang*, *Lang* ou encore *laNg*.

La pratique courante est de nommer toutes les variables d'environnement uniquement en anglais, en majuscules avec éventuellement des tirets de soulignement « _ ».

Affectation rapide et héritage avec Bash

L'interpréteur de commandes **bash** nous permet de définir une ou plusieurs variables d'environnement et de lancer un processus enfant en une seule commande. Par exemple, pour définir les variables « LANG » et « FOO », puis lancer **gedit**, nous pouvons utiliser la commande :

```
LANG=he_IL.UTF-8 FOO=bar gedit
```

Remarque : En utilisant cette commande, les nouvelles valeurs sont uniquement assignées au processsus enfant, ici **gedit**. Les variables du shell conservent leurs valeurs d'origine. Ainsi, dans cet exemple, la valeur de « LANG » restera inchangée (sans doute fr_FR.UTF-8 pour vous) pour les commandes suivantes dans le terminal.

Un comportement similaire peut être obtenu avec d'autres terminaux en utilisant la commande **env**.

Variables d'environnement persistantes

Jusqu'à présent nous avons vu comment définir ou modifier une variable d'environnement **temporairement**, jusqu'à ce que la session se ferme.

Variables d'environnement au niveau session utilisateur

Pour définir une variable d'environnement de manière à ce qu'elle affecte l'ensemble de la session d'un utilisateur, il suffit de placer une commande la définissant dans l'un des fichiers cachés de script présent dans le répertoire personnel de l'utilisateur. Voici les fichiers les plus courants qui peuvent être utilisés :

- **~/.profile** - C'est probablement le meilleur endroit pour placer une définition de variable d'environnement. En effet, il est exécuté automatiquement par le gestionnaire de connexion lors du démarrage d'une session graphique, mais aussi lors du démarrage d'une session en mode console texte.
- **~/.bash_profile** ou **~/.bash_login** - Si l'un de ces fichiers existe, il sera exécuté par **Bash** préférentiellement à **~/.profile** lors d'une connexion sur une console. (Bash utilisera ~/.bash_profile de préférence à ~/.bash_login). Cependant, ces fichiers n'auront par défaut aucune influence sur une session en mode graphique.
- **~/.bashrc** - Du fait de la manière dont Ubuntu configure par défaut les divers fichiers de

scripts, c'est sans doute l'endroit le plus facile pour définir des variables. La configuration par défaut garantit a peu près que ce fichier sera exécuté à chaque invocation de *bash* ainsi que lors de la connexion à l'environnement graphique. Cependant du point de vue des performances, ce n'est pas l'idéal car les variables seront inutilement redéfinies à chaque fois. (NdT : à chaque fois que vous ouvrez un terminal par exemple?)

Variables d'environnement au niveau système

Les variables d'environnement qui affectent l'ensemble du système (plutôt qu'une session utilisateur particulière) peuvent être définies dans l'un des nombreux scripts au niveau système, qui s'exécutent lors du chargement du système ou de l'environnement graphique. Ces définitions peuvent être placées dans plusieurs fichiers sur Ubuntu :

- **/etc/profile** - Ce fichier est exécuté quelle que soit la méthode de connexion utilisée : une console, une connexion distante ssh, ou une connexion en mode graphique. C'est probablement l'emplacement qui vous sera conseillé par les vieux routiers d'UNIX pour gérer vos variables d'environnement. Cependant sur Ubuntu ce script fait quelques vérifications puis invoque **/etc/bash.bashrc**.
- **/etc/bash.bashrc** - C'est la version au niveau système du fichier utilisateur **~/.bashrc**. Par défaut Ubuntu est configuré pour exécuter ce fichier quelle que soit la méthode de connexion, sur une console ou en environnement graphique.
- **/etc/environment** - Ce fichier est spécialement conçu pour recevoir les définitions de variables d'environnement au niveau système. Ce n'est pas un fichier de script, mais plutôt un fichier de déclarations de variables, ligne par ligne. En particulier, il contient les définitions des variables de langue et de la variable **PATH** au niveau système.

Remarque : sur des systèmes destinés à un usage personnel, il est sans doute préférable de définir les variables au niveau utilisateur tel que décrit plus haut, plutôt qu'au niveau système. En effet les fichiers utilisateurs peuvent être modifiés sans nécessiter les privilèges d'administration contrairement à ceux-ci.

Liste des variables d'environnement communes

Chaque application peut définir et utiliser ses propres variables d'environnement. De nombreuses pages de manuel contiennent de longue listes de variables pouvant affecter le comportement de l'application qu'elles décrivent. Cependant, les variables les plus utiles sont communes à de nombreuses applications.

Variables liées aux emplacements de fichiers

Les variables suivantes permettent au système de savoir où se trouvent divers fichiers, pour pouvoir fonctionner.

Variable	Exemples de valeur	Rôle
PATH	/usr/sbin:/usr/bin:/sbin:/bin	Lorsque vous tapez une commande, le système la cherche dans les dossiers spécifiés par la variable PATH, dans l'ordre où ils sont indiqués.
MANPATH	/usr/share/man:/usr/local/man	Liste de dossiers où le système doit chercher les pages de manuel.
LD_LIBRARY_PATH	/opt/app/oracle/lib	Liste de dossiers où le système doit chercher les bibliothèques d'exécution en plus de celles définies dans *ld* et */etc/ld.so.conf*
TMPDIR	/var/tmp	Le dossier utilisé pour les fichiers temporaires créés par de nombreux programmes

Variables de paramètres régionaux

Les variables d'environnement suivantes déterminent le comportement du système vis-à-vis de la langue et de la région, comme la langue utilisée pour envoyer des messages à l'utilisateur, ou le format de la date et de l'heure.

Les valeurs qui peuvent être assignées à ces variables d'environnement correspondent aux paramètres régionaux installés sur le système. Pour voir quels sont ces paramètres installés, vous pouvez utiliser la commande **locale -a**. les paramètres régionaux peuvent être générés par la commande **locale-gen**. Cependant, Ubuntu inclut des paramètres régionaux prédéfinis dans les paquets de langue (language-pack-xx) disponibles au travers du système de gestion de paquets

Variable	Rôle
LANG	Le paramètre linguistique de base utilisé par les applications du système, tant qu'il n'est pas contredit par une autre variable
LC_CTYPE	Le jeu de caractères utilisé pour saisir et afficher du texte
LC_NUMERIC	Mise en forme des valeurs numériques non-monétaires
LC_TIME	Format de la date et de l'heure
LC_COLLATE	Comment trier diverses informations, définit par exemple l'ordre alphabétique afin que les éléments puissent être triés alphabétiquement en utilisant la commande **sort**
LC_MONETARY	Format des valeurs numériques monétaires
LC_MESSAGES	Langue utilisée pour afficher les messages à l'utilisateur
LC_PAPER	Définitions des formats de papier standard
LC_NAME	Format des noms
LC_ADDRESS	Format des adresses
LC_TELEPHONE	Structure des numéros de téléphone
LC_MEASUREMENT	Unités de mesure à utiliser
LC_IDENTIFICATION	
LC_ALL	Cette variable a un rôle puissant pour écraser les autres paramètres régionaux. Lorsqu'une valeur lui est affectée, les applications utiliseront cette valeur quelle que soient les valeurs des autres variables

En utilisant diverses combinaisons pour ces variables de paramètres régionaux, vous pouvez obtenir des modifications intéressantes du comportement de votre système. Par exemple, votre système peut afficher les messages en anglo-américain (US-English), tout en utilisant les formats de date, nombres et unités de mesure les plus courants en Europe.

Ces variables peuvent prendre le pas les unes sur les autres dans certaines combinaisons. L'examen de la valeur des variables elles-mêmes ne fournit donc pas forcément une information claire sur la manière dont le système se comportera. La commande **locale** peut être utilisée pour examiner les valeurs effectives de ces variables pour les applications.

Variables pour les applications préférées

Ces variables d'environnement indiquent à divers programmes quelles sont les applications préférées pour effectuer certaines tâches.

En général, ces variables ne sont pas respectées par les applications en mode graphique qui intègrent leur propres éditeurs et afficheurs de texte. La plupart des environnements de bureau proposent en outre leur propre système de sélection des applications préférées.

Variable	Exemple de valeur	Rôle
PAGER	/usr/bin/less	Le nom de l'application utilisée pour afficher des textes longs (sur plusieurs pages écran) par des commandes telles que **man**
EDITOR	usr/bin/nano	Le nom de l'éditeur de texte préféré pour les utilisateurs. Il sera utilisé par des commandes telles que **mutt** ou **sudoedit**
VISUAL	/usr/bin/gedit	A le même rôle que la variable « EDITOR » mais est prioritaire. Si elle n'a pas de valeur « EDITOR » définie la valeur de l'application qui sera utilisée.
BROWSER	/usr/bin/lynx	Le nom du navigateur Web préféré pour les utilisateurs

Variables liées à l'environnement graphique

Variable	Exemple de valeur	Rôle
DISPLAY	:0.0 localhost:10.0 terminal01:0.0	Cette variable est utilisée pour indiquer aux applications où afficher l'interface graphique utilisateur. Sa valeur est constituée de trois parties : un nom d'hôte suivi de deux-points (:), un numéro d'affichage suivi d'un point (.) et un numéro d'écran. Le nom d'hôte peut être utilisé pour déporter l'affichage sur une machine distante du réseau. Il peut être omis si l'affichage se fait sur la machine locale. Le numéro d'affichage permet de choisir un parmi plusieurs serveurs X tournant sur la même machine (Ubuntu utilise plusieurs serveurs X pour permettre plusieurs sessions graphiques simultanées). Le numéro d'écran permet de choisir parmi plusieurs moniteurs gérés par le même serveur X. Sa valeur est généralement 0, et il est rarement utile de la modifier manuellement car cela peut être réglé automatiquement et intelligemment par de nombreuses applications comme **gdm** ou **ssh** lorsque c'est nécessaire.
XDG_DATA_HOME	~/.local/share	Indique aux applications conformes aux spécifications freedesktop.org, où placer les données de l'utilisateur. En général cette variable n'est pas définie puisqu'une valeur de secours par défaut est implémentée dans les spécifications.

Variable	Exemple de valeur	Rôle
XDG_CONFIG_HOME	~/.local/share	Indique aux applications conformes aux spécifications freedesktop.org, où placer les informations de configuration de l'utilisateur. En général cette variable n'est pas définie puisqu'une valeur de secours par défaut est implémentée dans les spécifications
XDG_DATA_DIRS	/usr/local/share:/usr/share	Une liste de dossiers séparés par deux-points (similaire à **PATH**) indiquant aux applications conformes aux spécifications freedesktop.org, où chercher les données. En général cette variable n'est pas définie puisqu'une valeur de secours par défaut est implémentée dans les spécifications
XDG_CONFIG_DIRS	/etc/xdg	Une liste de dossiers séparés par deux points (similaire à **PATH**) indiquant aux applications conformes aux spécifications freedesktop.org, où chercher les informations de configuration. En général cette variable n'est pas définie puisqu'une valeur de secours par défaut est implémentée dans les spécifications
XDG_CACHE_HOME	~/.cache	un emplacement utilisé par les applications conformes aux spécifications freedesktop.org, pour mettre en cache les données temporaires. En général cette variable n'est pas définie puisqu'une valeur de secours par défaut est implémentée dans les spécifications

Les variables XDG_* sont définies pour chaque utilisateur dans le fichier ~/.config/user-dirs.dirs. Leurs valeurs par défaut sont récupérés dans le fichier /etc/xdg/user-dirs.defaults.

Pour redéfinir ces variables utilisez une commande du type :

```
xdg-user-dirs-update --set DIR path
```

Remplacez DIR par un nom de variable présent dans /etc/xdg/user-dirs.defaults et path par le dossier de votre choix.

Par exemple pour redéfinir le dossier du bureau :

```
xdg-user-dirs-update --set DESKTOP $HOME/toto
```

Redéfinir (ou recréer) les répertoires par défaut

Si vous avez une ancienne installation d'Ubuntu que vous mettez à jour, il est probable que vous gardiez les anciens répertoires (Desktop pour le bureau, pas de répertoire Téléchargements, etc).

Pour forcer la redéfinition des répertoires, il faut lancer la commande suivante :

```
xdg-user-dirs-update --force
```

Variables spécifiques à GNOME

Variable	Exemple de valeur	Rôle
NAUTILUS_SCRIPT_SELECTED_FILE_PATHS	/home/ifireball/about.html	Cette variable d'environnement est définie par **nautilus**, le gestionnaire de fichiers de GNOME, lorsqu'un script est appelé à l'aide d'un clic droit. Elle consiste en une liste, ligne par ligne, des fichiers actuellement sélectionnés. Cette variable ne sera définie que s'il s'agit de fichiers locaux, ne provenant pas d'un partage réseau ou d'une connexion ssh.
NAUTILUS_SCRIPT_SELECTED_URIS	file:/home/ifireball/about.html	Cette variable d'environnement est définie par **nautilus**, le gestionnaire de fichiers de GNOME, lorsqu'un script est appelé à l'aide d'un clic droit. Elle consiste en une liste, ligne par ligne, des adresses (URI) des fichiers actuellement sélectionnés.
NAUTILUS_SCRIPT_CURRENT_URI	file:/home/ifireball	Cette variable d'environnement contient l'adresse (URI) de l'emplacement actuellement affiché dans la fenêtre de **nautilus** lorsqu'un script est appelé à l'aide d'un clic droit.
NAUTILUS_SCRIPT_WINDOW_GEOMETRY	828×511+251+342	Cette variable d'environnement contient la position à l'écran de la fenêtre de **nautilus** lorsqu'un script est appelé à l'aide d'un clic droit.

Variables d'exécution de programmes

La variable d'environnement suivante est incontestablement la plus puissante mais également la plus **dangereuse**. Elle permet de modifier la manière dont les applications s'exécutent.

Variable	Exemple de valeur	Rôle
LD_PRELOAD	/usr/lib/valgrind.so	Cette variable peut être utilisée pour injecter une bibliothèque dynamique personnalisée lors du chargement de l'application en mémoire. Cela peut servir à certaines choses, comme remplacer la bibliothèque d'allocation mémoire intégrée à l'application par une version de débogage, afin de détecter des fuites mémoire.

Variables de compilation

Variable	Exemple de valeur	Rôle
CC	gcc	Le nom du compilateur C à utiliser
CFLAGS	-o out.o	Une liste d'arguments de débogage / optimisation à passer au compilateur C
CXXFLAGS	-Wall	Une liste d'arguments de débogage / optimisation à passer au compilateur C++
CPPFLAGS	-DDEBUG	Une liste d'arguments à passer au préprocesseur / compilateur C/C++
LIBRARY_PATH	/usr/lib/firefox	Une liste de dossiers (séparés par deux-points) où chercher les fichiers de bibliothèques
INCLUDE	/opt/app/src/include	Une liste de dossiers (séparés par deux-points) où chercher les fichiers d'en-têtes
CPATH	..:$HOME/include:/usr/local/include	Une liste de dossiers (séparés par deux-points) où chercher les fichiers d'en-têtes

Autres variables d'environnement.

Variable	Exemple de valeur	Rôle
USERNAME	nom_utilisateur	Le nom de l'utilisateur actuellement connecté. Cette variable est définie par le système. Vous ne devriez certainement pas changer sa valeur manuellement.
LOGNAME	nom_utilisateur	Similaire à **USER**, certains programmes utilisent celle-ci de préférence à **USER** .
HOME	/home/utilisateur	Emplacement du répertoire personnel de l'utilisateur actuellement connecté.
PWD	/home/utilisateur/Desktop	Le répertoire de travail courant de l'interpréteur de commande.
SHELL	/bin/bash	L'interpréteur de commande préféré de l'utilisateur tel qu'il est défini dans le fichier « /etc/passwd » .
POSIXLY_CORRECT	s.o.	L'existence même de cette variable d'environnement, indépendamment de sa valeur, oblige quelques utilitaires à se comporter de manière plus conforme au standard POSIX. Cela peut provoquer le dysfonctionnement de divers outils GNU qui facilitent la vie, mais c'est peut être justement ce qui est recherché pour faire fonctionner correctement de vieux scripts UNIX.
HOSTALIASES	/etc/host.aliases	Le nom du fichier contenant les alias des noms d'hôtes pour utiliser avec divers logiciels réseau.
TZDIR	/usr/share/zoneinfo	Le chemin du dossier contenant les fichiers d'informations sur les fuseaux horaires (TimeZoneDIRectory). Il est généralement inutile de la définir manuellement car les systèmes cherchent de tels fichiers dans /usr/share/zoneinfo, par défaut.

Variable	Exemple de valeur	Rôle
TZ	IST-2 :Japan	Cette variable était utilisée par les anciens systèmes Unix pour spécifier le fuseau horaire du système. Cependant, Ubuntu et la plupart des systèmes récents utilisent pour cela le fichier /etc/localtime. Elle peut néanmoins être utilisée afin qu'une session utilisateur particulière affiche l'heure pour un fuseau horaire différent de celui du système. La valeur de cette variable peut être soit le nom et le décalage d'un fuseau horaire (premier exemple), soit le nom d'un fichier de zone situé dans /usr/share/zoneinfo (second exemple).
TERM	xterm	Le nom d'un fichier d'informations sur le terminal situé dans /lib/terminfo, ce fichier indique au programme en mode console comment effectuer certaines tâches telles que l'affichage des couleurs. Il peut être utile de jouer avec cette variable si vous essayez d'utiliser un programme d'émulation de terminal inhabituel, ou si vous essayez de connecter un émulateur de terminal matériel par le port série et que vous n'obtenez pas les résultats escomptés.
TERMCAP		Cette variable peut être utilisée à la place de “TERM” pour spécifier les informations sur le terminal manuellement, plutôt que d'avoir recours à un fichier.
COLUMNS	80	Le nombre de colonnes sur la fenêtre de console. Essayez d'ajuster cette variable si les lignes semblent ne pas être coupées correctement sur la console.
LINES	25	Le nombre de lignes sur la fenêtre de console. Essayez d'ajuster cette variable si vous obtenez des résultats étranges lorsque vous faites défiler le texte.
http_proxy	http://user:passwd@proxy:port/	La variable pour utiliser un proxy, marche aussi avec HTTP_PROXY, ftp_proxy et FTP_proxy .

Cette page est une adaptation en français de la page
https://help.ubuntu.com/community/EnvironmentVariables