Internationalisation et localisation des shell-scripts

Présentation

Objet, domaine concerné et public visé

Ce document a pour but d'aider les développeurs, mainteneurs et traducteurs à écrire/maintenir/traduire des shell-scripts en utilisant les outils disponibles dans GNU gettext.

Le document de référence est le manuel intitulé GNU 'gettext' utilities.

Le manuel envisage tous les langages de programmation qui peuvent être employés avec gettext, et fait principalement référence au langage C.

Voir POSIX specification qu'il est recommandé de lire, en particulier les volumes Base Definitions et Shell and Utilities.

À la différence du manuel, la portée du présent document se limite aux shell-scripts.

Mode opératoire

Le but est que les messages (habituellement des chaînes de caractères-texte) en sortie des shell-scripts soient affichés sur l'écran de l'utilisateur dans la langue de son choix.

L'utilisateur fixe ses préférences qu'il ou elle note dans la variable d'environnement LANG ou LANGUAGE (cette dernière enregistre une liste ordonnée suivant la priorité, de langues pour l'affichage des messages).

L'internationalisation (abrégée en I18N) est l'action qui consiste à:

- cocher les shell-scripts d'où proviennent les messages logiciels à traduire,
- puis à l'aide des outils de gettext élaborer un catalogue des messages-type à partir des scripts cochés.

Un catalogue de messages-type est souvent nommé “Portable Object Template” ou fichier POT.

Un fichier POT, lisible par les humains est, pour l'essentiel, constitué de chaînes de texte compactes, précédées de “msgid” pour “identifiant du message”, chaque chaîne étant suivie de “msgstr” et pour terminer, de la chaîne de texte qui est la traduction correspondante.

La localisation (abrégée en L10N) se fait:

- en générant depuis le fichier POT un “Portable Object” ou fichier PO pour chaque langue de destination,
- en écrivant les messages traduits dans les chaînes “msgstr” de chaque fichier PO,
- en vérifiant ces fichiers PO après traduction,
- en compilant chaque fichier PO pour obtenir un “Machine Objet” ou fichier MO.

Les fichiers MO qui, comme leur nom l'indique, sont lisibles seulement par la machine, sont

habituellement enregistrés en:

```
/usr/share/locale/<locale>/LC_MESSAGES/<software name>.mo
```

Dans le chemin d'accès ci-dessus, <locale> est un code de locale au format <ll[_TT]>, où ll est le code en deux lettres de la langue déterminée conformément au standard ISO 639-1 et le code TT, optionnel, les deux lettres indiquant le code-pays défini dans l'ISO 3166 pour la locale concernée; –exemple: fr_FR.

Chaque script corrigé devra comporter la commande suivante:

```
export TEXTDOMAIN=<nom du logiciel>
```

Pendant l'exécution du script cela permet à “gettext” d'atteindre le fichier MO qui convient pour afficher chaque message coché, dans la langue choisie telle que l'indique la variable d'environnement LANG ou LANGUAGE.

Diagrammes des processus

Soit un logiciel donné avec l'ensemble des shell-scripts qu'il contient et que nous voulons internationaliser et localiser.

Les diagrammes suivants donnent une vue d'ensemble de chacun des processus concernés: internationalisation, localisation, usage et maintenance.

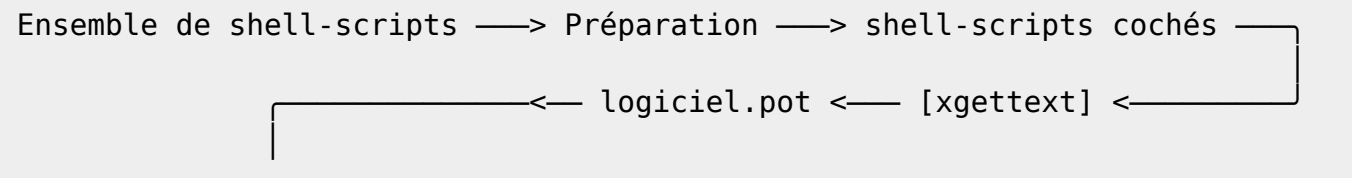
Ces diagrammes sont hybrides, i.e. ils présentent des données aussi bien que des actions.

Parmi ces actions, il y a l'exécution de plusieurs programmes de la suite 'gettext':

- gettetxt: marque les libellés à internationaliser, puis affiche les messages localisés pendant l'exécution des scripts
- xgettext: extrait les libellés choisis d'un ensemble de shell-scripts pour créer un fichier POT ou PO
- msgcmp: pour vérification de leur cohérence, compare un fichier PO à un autre, ou à un fichier POT
- msginit: écrit un fichier PO à partir d'un POT donné en entrée
- msgfmt: formate un fichier MO à partir d'un fichier POI en entrée
- msgmerge: fusionne ou met à jour les fichiers PO ou POT

Dans les diagrammes ci-dessous, les programmes “gettext” apparaissent entre crochets.

(1) Internationalisation



```
                ├─────> (2) Localisation
                │
                └─────> (4) Maintenance
```

(2) Localisation (exemple pour le français et le néerlandais).

```
     ┌────────────────<── logiciel.pot <── (1) Internationalisation
     │
     │
     ├──> [msginit] ──> fr_FR.po ──> éditeur de PO ──> fr_FR.po ──> [msgcmp]
──┐
     │
│
     │       ┌─ installation <── fr_FR.mo <── [msgfmt] <─┬─ fr_FR.po vérifié
<─┘
     │       │                                           │
     │       │                                           └─────> (4)
Maintenance
     │       │
     │       └─> /usr/share/locale/fr_FR/LC_MESSAGES/logiciel.mo ─> (3)
Usage
     │
     └──> [msginit] ──> nl_NL.po ──> éditeur de PO ──> nl_NL.po ──> [msgcmp]
──┐
│
             ┌─ installation <── nl_NL.mo <── [msgfmt] <─┬─ nl_NL.po vérifié
<─┘
             │                                           │
             │                                           └──────────> (4)
Maintenance
             │
             └──> /usr/share/locale/nl_NL/LC_MESSAGES/logiciel.mo ─> (3)
Usage
```

(3) Usage

Prenons le cas où un des scripts, “myscript.sh” inclut cette commande:

```
gettext "Good morning"
```

et que “Good morning” est traduit comme suit dans les catalogues de messages:

```
/usr/share/locale/fr_FR/LC_MESSAGES/PACKAGE.mo ─> "Bonjour"
/usr/share/locale/nl/LC_MESSAGES/PACKAGE.mo ─> "Goedemorgen"
```

Voici ce que verra l'utilisateur en fonction du contenu de la variable globale LANG:

```
             ┌────────────────<── (2) Localisation
```

```
                 │
 LANG=fr_FR ├──────> sh myscript.sh or ./myscript.sh ──────> "Bonjour"
                 │
 LANG=nl_NL └──────> sh myscript.sh or ./myscript.sh ──────> "Goedemorgen"
```

(4) Maintenance

Le processus de maintenance peut être déclenché par la création, la modification ou l'effacement d'un script.

Dans le diagramme ci-dessous, l'étape du processus qui débute par la commande msmerge devrait être répétée pour chaque fichier PO présent.

Il est donc judicieux de conserver une liste à jour des traductions disponibles sous forme de fichiers PO.

```
        Shell-scripts mis à jour et cochés ──────> [xgettext] ──────> logiciel.pot
──────┐
      │
                                          (1) Localisation ──> <locale>.po
──────>┬<──┘
      │
      ┌─ [msgcmp] <── <locale>.po <─ éditeur de PO <── <locale>.po <─ [msmerge]
<──┘
      │
      └──> <locale>.po vérifiée -> [msgfmt] -> <locale>.mo -> installation  ──────┐
                                                                                  │
                         /usr/share/locale/<locale>/LC_MESSAGES/logiciel.mo <──────┘
```

Le processus de maintenance peut également être lancé par la modification d'un catalogue de messages pour une langue en particulier (pour corriger une erreur par exemple).

Cette variante du processus est plus courte:

```
┌─ [msgcmp] <── <locale>.po <── éditeur de PO <── <locale>.po <── mise à
jour nécessaire
│
└─> <locale>.po vérifiée ──> [msgfmt] ──> <locale>.mo ──> installation  ──┐
                                                                          │
                          /usr/share/locale/<locale>/LC_MESSAGES/logiciel.mo <──┘
```

Processus d' internationalisation

Ce chapitre concerne les développeurs et les mainteneurs.

Le processus d'internationalisation comprend les tâches suivantes:

1. Préparer les scripts à internationaliser
2. Repérer les messages à localiser
3. Utiliser 'xgettext' pour produire un catalogue modèle de messages

Préparer les scripts à l'internationalisation

Cette tâche est nécessaire pour les scripts qui ne présentent pas encore toutes les cararctéristiques permettant l'internationalisation.

Note technique: conditions que Gettext requiert pour les shell-scripts.

La liste suivante de conditions requises n'est pas complète.

Elle comprend seulement les recommandations principales basées sur mon expérience et que le développeur ou le mainteneur auront à vérifier. Gettext remplace à l'exécution les chaînes de caractères provenant de:

- une commande “echo” ou
- un programme (tel que 'dialog', par exemple)

par les chaînes-texte (trouvées dans un catalogue de messages pour la langue déterminée par $LANG or $LANGUAGE).

Mais le remplacement ne se fait que si les conditions suivantes sont remplies:

- Qu'un fichier MO soit accessible par le chemin d'accès établi à partir de la variable d'environnement TEXTDOMAIN, écrit ainsi:

<dir_name>/<locale>/LC_MESSAGES/text_domain.mo.
Par exemple, si TEXTDOMAIN=le_logiciel et $LANG=de_DE.utf8, gettext cherchera:
<dir_name>/de_DE/LC_MESSAGES/software.mo
<dir_name> peut être déterminé grâce à la valeur de la variable d'environnement TEXTDOMAINDIR sinon on emploie une valeur par défaut qui est pour Slackware Linux: /usr/share/locale.
Il y a des difficultés, par exemple si <locale> est “de_DE”, le fichier mo pourrait se trouver en <dir_name>/de/LC_MESSAGES/ au lieu de <dir_name>/de_DE/LC_MESSAGES/

- Que la variable TEXTDOMAIN soit exportée avant tout lancement d'une commande *gettext.
- Que gettext.sh, qui fournit les fonctions eval_gettext et eval_ngettext, soit renseigné avant un appel de ces fonctions.
- Qu'une chaîne msgid dans le fichier MO corresponde exactement à l'argument de gettext (ou de eval_gettext s'il y a dans la chaîne un paramètre d'expansion).
- Qu'il n'y ait pas une barre oblique inversée suivie d'un espace dans la chaîne msgstr correspondante.
- Que la chaîne msgstr commence et finisse par 'newline' ou non, à l'dentique de msgid.
- Que dans le cas où la chaîne de texte contient un paramètre d'expansion, eval_gettext soit employé à la place de gettext.
- “The variable names must consist solely of alphanumeric or underscore ASCII characters, not start with a digit and be nonempty; otherwise such a variable reference is ignored.” (nous rappelle le manuel de gettext): Les noms de variables ne doivent comporter uniquement que des caractères alphanumériques ou le tiret bas, ne peuvent pas commencer par un chiffre ni

être une chaîne vide; sinon la référence à la variable (son nom) est ignorée.
- Que l'évitement des paramètres d'expansion soit obtenu par une seule barre oblique inversée, comme ceci:
  \$parameter or \${parameter}
  sauf si la commande eval_gettext se trouve dans une commande de substitution comme celle-ci:
  “`eval_gettext ”...“`” ou “$(eval_gettext ”...“)“
  Dans ce cas, il faut mettre trois barres obliques inversées, comme ceci:
  \\\$parameter or \\\${parameter}.
- Que seules les formes $parameter and ${parameter} du paramètre d'expansion soient employées dans un argument d'eval_gettext (toute autre forme est interdite).
- Que les paramètres positionnels, paramètres spéciaux et commandes de substitution ***ne soient pas*** employés dans un argument de gettext ou d'eval_gettext.

Une conséquence pratique des deux dernières règles conduit à recommander que tous les paramètres positionnels, paramètres spéciaux, les commandes de substitution et les formes non autorisées des paramètres de substitution soient assignées en amont aux variables nommées, puis développées dans la chaîne de texte en argument d'eval_gettext ou eval_ngettext.

Conseil: si une chaîne de texte a été incluse en tant que msgid dans un catalogue de messages et est assignée à une variable nommée dans un script, alors les commandes: “gettext $parameter” et “gettext ${parameter}” donneront en sortie la chaîne traduite à l'exécution, et d'ailleurs 'xgettext' mettra de côté cette commande en suivant le script, parce que 'gettext' est employé au lieu de 'eval_gettext'. Cela peut être commode. Dans ce cas le caractère d'expansion n'aura pas à être “échappé”.

Marquer les messages à localiser

Je recommande de marquer les messages qui sont:

- les arguments d'une commande 'echo' non redirigée
- les arguments de commandes 'echo' redirigées, chaque fois qu'un traitement ultérieur les affiche sur l'écran de l'utilisateur
- les arguments d'autres commandes qui affichent le message, par exemple le programme 'dialog'

Par contre je recommande de ne pas marquer:

- les commentaires destinés aux lecteurs du script,
- les chaînes de texte dont la valeur sera traitée plus tard, par exemple en tant qu'arguments d'une commande composée 'case', ou les <tag> qui servent à la commande d'un menu de dialogue.

Parfois un shell-script écrit d'autres shell-scripts.

Sur ces bases, c'est au développeur ou au mainteneur de décider au cas par cas ce qui doit être marqué en fonction des intentions fixées pour l'internationalisation.

Se servir de 'xgettext' pour produire un modèle de catalogue des messages

Il faut choisir entre la création d'un seul fichier POT pour le logiciel tout entier ou bien d'un fichier POT par groupe de scripts, en envisageant par exemple quelle option minimisera le travail de maintenance, comment le travail de localisation sera organisé, le niveau de fréquence relative des mises à jour pour les différents ensembles de scripts que comprend le logiciel, et qu'il est pertinent de distinguer les groupes de fonctionnalités concernant l'installation qui diffère de la configuration, distincte de la gestion des paquets.

J'ai pour ma part tendance à faire un seul fichier POT, à vous de choisir.

Si le logiciel comporte plusieurs scripts situés en différents endroits ou inclus dans plusieurs paquets, il peut être utile de regrouper une copie de tous les scripts dans un seul répertoire, et/ou enregistrer dans un fichier texte une liste de tous ces scripts avec leurs chemins d'accès.

Le fichier POT sera produit en utilisant la commande 'xgettext' (voir le manuel ou 'xgettext –help' pour les détails).

Options à inclure dans la commande:

```
-L Shell (évidemment!)
--strict (pour faciliter les vérifications et la gestion des catalogues de
messages)
-c       (pour inclure des commentaires utiles aux traducteurs dans le
fichier POT)
-n       (pour identifier le source et le n° de ligne de chaque message
         --Ce sont les options minimales par défaut.)
```

Une fois que le fichier POT est obtenu vous pourrez vérifier qu'il inclut les entrées pour tous les appels à *gettext dans le(s) script(s).

Processus de localisation

Quand le fichier POT est disponible, la commande 'msginit' écrit un fichier PO pour chaque langue de destination.

Dans les fichiers PO les chaînes “msgid” ne doivent jamais être modifiées, sinon la traduction ne se ferait pas à l'exécution.

La commande 'msgcmp' permet de vérifier chaque fichier PO par rapport au POT après traduction, afin de s'assurer que tous les messages sont traduits.

Le traducteur peut employer la commande 'msgfmt' pour vérifier la composition du texte traduit.

Il faut prendre soin de sauvegarder le fichier PO car on peut en avoir besoin plus tard pour la maintenance (il est encore possible de recréer un fichier PO en appliquant la commande 'msgunfmt' à un fichier MO mais le contexte pourrait avoir été perdu, ce qui rendrait le fichier PO reconstitué presque inutilisable).

Le fichier PO vérifié est remis au mainteneur qui, en lançant 'msgfmt' obtient le fichier MO, puis

l'installe.

Processus d'utilisation

La seule chose que l'utilisateur doit faire est d'indiquer sa ou ses langue(s) préférée(s).

Ce qui se fait essentiellement en renseignant la variable d'environnement LANG.

On peut le faire juste avant de lancer le script en écrivant d'abord LANG=<locale>, mais habituellement l'utilisateur précise son choix de langue(s) de façon permanente.

Ce qui se fait dans Slackware Linux en éditant le ou les fichier(s) /etc/profile.d/lang.sh et/ou /etc/profile.d/lang.csh (voir ces fichiers).

Les modifications seront actives au prochain redémarrage.

Je suggère d'utiliser une locale UTF-8, comme pour lire ce document.

Si l'utilisateur emploie plusieurs langues, une autre option est d'indiquer dans la variable d'environnement spécifique à gettext LANGUAGE une liste de langues classées par ordre de priorité.

Par exemple, si LANGUAGE est établie à 'de:fr', une traduction en Allemand sera utilisée si elle est disponible, sinon une traduction en Français sera utilisée si elle est disponible, et sinon les messages seront affichés dans la langue d'origine, souvent l'Anglais. Voyez le manuel de gettext pour les détails.

Processus de maintenance

Dans la majorité des cas le processus de maintenance sera déclenché par la création d'un script, sa modification ou sa suppression.

Dans un tel cas le mainteneur créera un nouveau fichier POT avec 'xgettext' pour le transmettre aux traducteurs.

Les traducteurs utiliseront le nouveau fichier POT pour mettre à jour leurs fichiers PO respectifs (sauvegardés), avec la commande 'msgmerge –update'.

Puis ils éditeront/compléteront les traductions en portant toute leur attention sur les messages pas encore traduits et sur ceux annotés “imprécis” dans les fichiers PO, en utilisant un éditeur de PO.

Après cela le fichier PO sera vérifié et comparé au fichier POT avec 'msgcmp', il sera soigneusement enregistré, transmis au mainteneur qui fera créer le nouveau fichier MO par 'msgfmt' et l'installera comme dans le processus de localisation initial.

Le processus de maintenance déclenché par une modification nécessaire d'un fichier PO pour une langue déterminée est similaire, mais plus court: il commencera par la mise à jour du fichier PO correspondant, par le traducteur. Pour réduire la charge de travail que ce type de maintenance entraîne, je suggère que le mainteneur demande qu'on lui adresse uniquement des traductions

complètes et bien révisées.

Recommandations pratiques aux développeurs et aux mainteneurs

En Anglais beaucoup de mots sont polysémiques: leur sens ne peut être déterminé que d'après le contexte de leur emploi. Dans la pratique il en résulte que plus vous précisez le contexte, plus la traduction peut être fidèle.

Exemple: récemment, pendant le téléchargement d'un logiciel, j'ai vu une indication du genre:
31min gauche
Vous voyez? Après j'ai compris que “left” avait été traduit par “gauche” (comme dans “left hand”).

Également, l'ordre des mots dans une phrase varie suivant la langue, en outre toutes les langues ne s'écrivent pas de gauche à droite. Donc, marquez des paragraphes entiers, ou au moins des phrases entières, pas des lignes, laissez seuls les mots isolés sauf dans des cas particuliers.

Par exemple, si des paragraphes de texte ont été scindés en lignes affichées par des commandes 'echo', remplacez toutes ces commandes 'echo' consécutives par une seule commande 'gettext' ou 'eval_gettext'.

Ne craignez pas d'inclure les substitutions de variables dans les phrases, l'éditeur de PO vérifiera qu'elles sont présentes telles quelles dans les traductions.

Recommendations pour le programme 'dialog'.

Le programme 'dialog' présente une Interface Utilisateur (UI) sous la forme de boîtes de dialogue.

D'autres programmes fournissent la même possibilité, pour ceux-là j'estime (ce n'est qu'une estimation) que les présentes recommendations peuvent s'appliquer aussi.

Gardez présentes à l'esprit les considérations suivantes, au moment de fixer ou de revoir les options de conception des boîtes de dialogue.

- Les messages traduits dans d'autres langues seront souvent bien plus longs que dans leur version originale (habituellement en Anglais).
- Dans les situations où on ne dispose que des pilotes VGA (comme dans les installateurs en mode texte), l'affichage à l'écran est généralement limité à 25 lignes de 80 colonnes avec les polices les plus répandues, mais en pratique le passage à la ligne peut se produire quand une ligne dépasse 74 caractères de long.
  Par conséquent, pour les affichages statiques la longueur des lignes de texte devrait être de 74 caractères au maximum.
- Le défilement vertical du texte est bien accepté car très souvent utilisé pour l'affichage des pages sur Internet, il est parfois inévitable.
  Au contraire, le défilement horizontal devrait autant que possible être évité.

Je suggère donc de:

- renoncer à bien ajuster les dimensions des boîtes à la taille du texte anglais puisque la traduction casserait probablement votre mise en page soigneusement élaborée, à moins d'imposer des contraintes excessives (IMO) aux traducteurs.

- en particulier, ne pas réduire la largeur des boîtes au strict nécessaire pour l'affichage des textes en Anglais, tout spécialement dans les mises en page de tableaux où le texte ne peut pas déborder sur les lignes suivantes,
- favoriser une mise en page fluide du texte affiché opposée à une présentation fixe, pour éviter des lignes trop longues dans les traductions, ce qui imposerait le défilement horizontal (qui n'est d'ailleurs pas toujours possible).

En particulier, je recommande de privilégier les options qui prennent comme premier argument une chaîne de texte à la place d'un fichier, pour permettre le retour à la ligne. On peut alors conserver la mise en page souhaitée en employant des espaces blancs pour l'indentation.

Par exemple,

```
dialog <common-options> --textbox <file> <height> <width>
```

peut être remplacé par

```
dialog --no-collapse <common-options> --msgbox "`cat <file>`" <height>
<width>
```

Conseils pratiques aux traducteurs

Suivant le volume de travail nécessaire et les ressources disponibles, il peut y avoir un traducteur ou bien une équipe de traducteurs par langue de destination. Dans tous les cas je recommande qu'au moins une personne ait la responsabilité d'organiser le travail de l'équipe, de vérifier les traductions et de transmettre le fichier PO au(x) mainteneur(s). Appelons cette personne: «coordinateur».

Il ne faut pas obligatoirement traduire mot à mot. Non seulement c'est rarement la meilleure façon de communiquer clairement le sens, mais encore cela produit souvent des phrases trop longues pour entrer dans l'espace prévu.

Utilisez un éditeur de PO spécifique, 'pas' un éditeur de texte standard. D'une part vous éviterez ainsi d'éditer des chaînes 'msgid' par inadvertance et d'autre part leur action en sera facilitée et les vérifications complémentaires seront automatisées, comme la présence d'une variable ayant la même orthographe dans la traduction et dans l'original.

Pendant la traduction, choisissez une police sérif de largeur fixe (ou “monospace”), comme Courier. Cela évite de confondre des caractères qui autrement se ressemblent, on peut aussi vérifier la longueur de ligne quand cela importe.

Si possible, vérifiez l'affichage des messages. Vous pourriez le faire en regardant le contexte dans le fichier source correspondant. Mieux encore, activez tout simplement le script traduit.

Ceci est très important si vous traduisez des boîtes de dialogue. En particulier, évitez d'écrire des phrases trop longues sur une seule ligne s'il s'avère que le texte ne peut pas continuer sur la ligne suivante.

N'oubliez pas qu'en mode VGA (utilisé dans les installeurs à interface en texte, par exemple), la dimension des lignes est limitée en théorie à 80 caractères, mais souvent à 74 en fait.

N'ajoutez pas de points d'interrogation qui ne figurent pas dans le message d'origine.

Si le message concerne des invites (un texte bref sur les boutons) de boîtes de dialogue, comme “OK”, “Yes”, “NO”, “Continue”, “Cancel”, assurez vous de la traduction dans votre langue de ces invites employées dans l'interface globale de dialogue et reprenez les mêmes mots.

Évitez les expressions familières et l'argot technique.

Pour “couper” (ou terminer) une ligne dans une boîte de dialogue vous entrerez \n: et quand il enfoncera [Enter], l'utilisateur 'ne verra pas' de caractère “new line” s'afficher dans le texte sous ses yeux.

En outre, vous devrez vous conformer aux exigences suivantes de gettext pour qu'il fonctionne:

- S'il y a un mot commençant par le signe dollar dans le texte d'origine, il devra apparaître dans la traduction avec exactement la même écriture (la casse compte).
- Le texte traduit doit comporter un caractère de passage à la ligne (ou “saut de ligne” représenté par '\n') au début ou à la fin, exactement comme dans le texte d'origine. S'il n'y a pas le caractère '\n' dans le texte d'origine, alors il ne doit pas être dans la traduction, non plus.
- Le caractère barre oblique inverse “\” n'est pas autorisé dans la traduction.

Pour vérifier que votre traduction satisfait aux exigences de gettext vous pourrez appeler cette commande :

```
msgfmt -c <nom du fichier PO>
```

Points importants pour traduire les pages de manuel

Conservez attentivement la syntaxe des pages de manuel telles qu'elles sont notées en anglais. Il ne faut pas changer:

- 'B<' en 'B <' (n'insérez pas l'espace)
- 'B<' en 'b<' (laissez le B en majuscule - et ne le remplacez pas par la lettre majuscule grecque BETA qui a le même aspect à l'écran)
- “I” en '|' (ne remplacez pas la lettre majuscule I par le symbole 'pipe' – alt gr + 6 –)

En traduisant des commandes du shell, laissez en anglais les noms de chemins d'accès à utiliser. Mais vous pouvez et devez traduire les arguments destinés à prendre une valeur comme 'packagename', dans ce cas –> 'nomdupaquet'.

Didier Spaier

Sources

* Originally written by Didier Spaier * Translation October 2015 by P-M Averseng

howtos, gettext, shell, scripts, internationalization, localization, i18n, l10n, translator pierreaverseng