Licence d'informatique
Module de C/Unix

La commande `loghdl`

Examen – Partie Unix

Philippe MARQUET

Janvier 2001

Les documents de cours et TD sont autorisés.
On rendra deux copies séparées pour la partie langage C et pour la partie Unix ; la partie Unix sur une *copie de couleur.*

1 Questions de cours

Un utilisateur exécute en interactif la commande suivante :

```
% (cd ../src && emacs & )
```

Question 1
Pour chacune des quatre constructions suivantes de cette commande, expliquez :

a. La signification de la construction

b. Pourquoi l'utilisateur utilise cette construction

Aucune de vos explications ne doit dépasser deux lignes.

1. … `&&` …
2. … `&`
3. `cd ../src`
4. `(` … `)`

2 Gestion de fichiers de log

Nous allons développer un outil de manipulation de fichiers de log : `loghdl` (*log handling*).

2.1 Message de log, fichier de log, jeu de fichiers de log

De nombreuses commandes d'administration système génèrent des *fichiers de log*. Ces fichiers contiennent des *messages de log*, c'est-à-dire une trace de l'exécution des commandes et/ou des messages d'erreurs. Ces fichiers sont consultés par l'administrateur système pour s'assurer du bon fonctionnement du système ou détecter les comportements suspects.

Cependant, il n'est pas envisageable de conserver indéfiniment l'ensemble des messages de log produits par le système : la place disque est limitée, et seuls les messages de log les plus récents sont régulièrement consultés par l'administrateur système.

Pour chaque fichier de log, l'administrateur système peut choisir de conserver un certain nombre des anciennes versions du fichier sous une forme tournante.

Un *jeu de fichiers de log* est constitué d'un fichier de log et de ses anciennes versions. Ces anciennes versions sont nommées d'après le nom du fichier de log suffixé par des entiers successifs. Ces anciennes versions peuvent aussi être compressées ; on trouvera alors des fichiers pourvus d'une extension indiquant ce fait. Les deux suites de noms suivantes peuvent donc désigner un jeu de fichiers de log :

Id : xam01.tex,v 1.6 2001/08/28 08 :51 :09 marquet Exp

1. `foo foo.1 foo.2 foo.3`
2. `bar bar.1.gz bar.2.gz bar.3.gz bar.4.gz`

À chaque invocation, la commande `loghdl` recherche les jeux de fichiers de log qui doivent être mis à jour. Un jeu de fichier peut avoir à être mis à jour si :

– le fichier de log est plus ancien (date de création) qu'une durée donnée ; ou
– le fichier de log occupe une place disque plus importante qu'une taille donnée.

La mise à jour d'un jeu de fichiers de log est réalisée sous la forme d'une rotation des fichiers au sein du jeu : la création d'un nouveau fichier de log vide entraîne la suppression de la version la plus ancienne.

2.2 Fichier de configuration de **`loghdl`**

Notre commande `loghdl` est paramétrée par un fichier de configuration. Un fichier de configuration est constitué d'une suite de sections.

Il existe une section générale et une section par fichier de log géré. La section générale est placée en début du fichier de configuration. Le début de la section relative au fichier de log *`log-file`* est marquée par une ligne

```
[log-file]
```

Une section est formée de lignes de la forme :

```
variable = value
```

Les lignes vides (i.e. ne comportant que des espaces) et les lignes dont le premier caractère significatif est un dièse # sont ignorées.

Un fichier de configuration produit deux informations :

1. La liste des fichiers de log géré.
2. La valeur d'un certain nombre de variables. Pour un fichier de log, la valeur d'une variable donnée est définie dans la section du fichier ; à défaut dans la section principale.

Exemple de fichier de configuration :

```
# Fichier de configuration pour loghdl

# A qui rapporter les erreurs
MAIL = sysadmin@lifl.fr
# La commande de compression
COMPRESS = gzip -best
EXTENSION = .gz
# Mise a jour a l'anciennete
TEST = AGE
AGE = 2 weeks
# 4 fichiers dans un jeu
COUNT = 4

[/var/log/http]
MAIL = sysadmin@lifl.fr webmaster@lifl.fr
COUNT = 8
AGE = 1 week

[/var/log/bootlog]
TEST = SIZE
SIZE = 500k

[/var/log/lpd-errors]
```

2.3 Variables de configuration

Les principales (et seules nécessaires pour traiter le présent sujet) variables définies dans un fichier de configuration sont :

`COMPRESS` indique le nom d'une commande qui remplace un fichier par une version compressée de celui-ci. Le nom du fichier compressé est celui du fichier originel suffixé par la valeur de la variable `EXTENSION`.

`EXTENSION` doit être positionnée en accord avec `COMPRESS`.

`COUNT` est le nombre de fichiers à conserver dans un jeu de fichiers de log.

`SIZE` est la taille d'un fichier de log au delà de laquelle le jeu de fichiers de log doit être mis a jour (si ce critère de mise à jour est sélectionné ; voir `TEST`).

`AGE` est l'âge d'un fichier de log (à compter de sa création, c'est-à-dire la dernière mise à jour du jeu de fichiers de log) au delà de laquelle le jeu de fichiers de log doit être mis à jour (si ce critère de mise à jour est sélectionné ; voir `TEST`).

`TEST` prend une des trois valeurs `AGE`, `SIZE`, ou `BOTH` pour indiquer que le critère d'âge (resp. de taille, resp. les deux) doit être pris en compte pour considérer une mise à jour du jeu de fichiers de log.

2.4 Calcul de durées et de dates, Calcul de taille

La commande `loghdl` doit manipuler des dates et des durées. Typiquement, savoir si un fichier donné est plus âgé qu'une durée donnée. De même la commande doit manipuler des tailles de fichier.

Dans le cadre de ce sujet, nous ne nous préoccupons pas de détails à propos de ces deux types de données.

Les durées exprimées dans un fichier de configuration de `loghdl` suivent la syntaxe illustrée sur l'exemple de la section précédente. De même pour les tailles.

Vous pouvez utiliser les deux commandes suivantes qui traitent des dates et tailles avec une syntaxe compatible avec `loghdl` :

- `biggerthan` *`log-file size`*
- `olderthan` *`log-file duration`*

et qui retournent un succès si et seulement si le fichier donné est plus gros (resp. plus ancien) que la taille (resp. date) donnée.

2.5 Commande `loghdl`

Il s'agit dans un premier temps de se fournir de petits outils sous forme de fonctions Bourne shell pour ensuite écrire la commande `loghdl`.

Dans la suite,

- *`cfg-file`* désigne un fichier de configuration.
- *`log-file`* désigne un fichier de log.
- *`var`* désigne une variable de configuration.

Question 2 (`get_logfiles`)
La fonction

```
get_logfiles cfg-file
```

affiche la liste des fichiers de log gérés par *`cfg-file`*.

Donnez un algorithme et le code Bourne shell commenté de la fonction `get_logfiles`.

Question 3 (`get_val`)
La fonction

```
get_val cfg-file log-file var
```

affiche la valeur de la variable *var* pour le fichier de log *log-file*.

Donnez un algorithme et le code Bourne shell commenté de la fonction `get_var`.

Question 4 (`get_fileset`)
La fonction

```
get_fileset cfg-file log-file
```

affiche l'ensemble des fichiers pouvant appartenir au jeu de fichiers de log du fichier *log-file*. Certains de ces fichiers peuvent ne pas exister (parce que le nombre de mise à jour sur ce jeu de fichiers n'est pas encore suffisant). Cette liste de noms est ordonnée, la version la plus ancienne de ce fichier en tête, le fichier de log en queue.

Donnez un algorithme et le code Bourne shell commenté de la fonction `get_fileset`.

Question 5 (`loghdl`)
La commande `loghdl` accepte en paramètre un unique argument qui est le nom du fichier de configuration à traiter :

```
loghdl cfg-file
```

Donnez un algorithme et le code Bourne shell commenté de la commande `loghdl`.

Fin du sujet