Licence d'informatique
Module de C/Unix

Projet Unix

`dotg` — Dot file generator

Philippe MARQUET

Octobre 2003

L'objet de la commande `dotg` (Dot file generator) que vous allez développer est de produire des fichiers de configuration tels `.login`, `.cshrc`, `.emacs` pour plusieurs systèmes donnés en fonction d'un fichier unique et des caractéristiques des différents systèmes.

1 Pourquoi `dotg` ?

Les fichiers de configurations tels `.login` ou `.emacs` permettent de modifier le comportement de certaines commandes ou outils.

La mise à jour de ces fichiers est parfois laborieuse quand on utilise plusieurs comptes. Par exemple un étudiant peut avoir un compte sur sa machine personnelle à son domicile et un compte sur les machines du réseau d'enseignement du M5... Les fichiers de configuration pour ces deux comptes vont être semblables sans être strictement identiques.

L'utilisation de `dotg`
- centralise les modifications de ces fichiers de configuration ;
- assure une cohérence, et permet donc de travailler dans un environnement semblable indépendamment du compte utilisé ;
- autorise la personnalisation de l'environnement en fonction du compte utilisé.

La commande `dotg` est un préprocesseur : elle transforme des fichiers DOTG en des fichiers de configuration.

Les fonctionnalités de base de `dotg` sont :
- la définition et l'expansion de macros ;
- l'inclusion de fichiers ;
- la production conditionnelle de lignes.

2 Commandes DOTG

Les commandes `DOTG` sont les suivantes :

`#include` permet l'inclusion d'un autre fichier source DOTG.

`#define` crée une nouvelle macro. Certaines macros spéciales conditionnent le comportement de `dotg`.

`#predicate` définit un nouveau prédicat qui sera utilisé avec les commandes `#if`

`#if/#ifnot/#endif` permettent la production conditionnelle de lignes de texte.

`#openfile/#closefile` marquent le début et la fin de production de texte.

`#info` génère une ligne de promotion pour `dotg`.

Une commande DOTG doit apparaître seule sur une ligne, toutes les commandes DOTG débutent par un caractère `#`.

Un fichier DOTG comporte des commandes et des lignes de texte parmi lesquelles on distingue :

Id: dotg.tex,v 1.4 2003/10/15 10:27:42 marquet Exp

– celles comprises entre un `#openfile` et un `#closefile`. Elles sont produites dans le fichier de configuration généré ;
– celles situées hors un bloc `#openfile/#closefile`. Ces ligens de texte sont ignorées.

La commande **#include**

Un texte commun à plusieurs fichiers DOTG peut être placé dans un fichier commun (par exemple `common.dotg`) et inclus dans chacun des fichiers par la commande

```
#include "common.dotg"
```

`dotg` remplace cette ligne par le contenu du fichier `common.dotg`.
Ce fichier peut contenir d'autres commandes DOTG.

La commande **#define** et les macros

La commande `#define` définit des macros qui sont des noms auxquels sont associés une valeur. Par exemple

```
#define OUTPUTDIR /users/phm
```

définit la macro `OUTPUTDIR` dont la valeur est « `/users/phm` ».
L'utilisation de cette macro, c'est-à-dire la substitution de son nom par sa valeur, est spécifiée dans le source DOTG par l'utilisation de chevrons :

```
<<OUTPUTDIR>>
```

D'autres choses sont possibles avec les macros :
– la valeur de macros peut être définie sur la ligne de commande de `dotg` (voir section 4) ;
– il existe un certain nombre de macros spéciales (voir section 3) ;
– il est parfaitement légal de trouver plusieurs `#define` pour une même macro. Il est ainsi possible de changer la valeur d'une macro lors du traitement de différentes parties de vos fichiers DOTG ;
– la valeur d'une macro peut elle même contenir le nom d'une autre macro, par exemple :

```
#define DotG_OUTPUTFILE <<OUTPUTDIR>>/.emacs
```

Notez que les définitions imbriquées sont évaluées lors de leur utilisation et non lors de leur définition. Ainsi, sur l'exemple, il est possible de modifier la valeur de la macro `OUTPUTDIR` après la définition de la macro `DotG_OUTPUTFILE`.
– l'utilisation d'une macro non définie provoque une erreur.

Les commandes **#openfile** et **#closefile**

Un couple de commande `#openfile/#closefile` encadre un bloc de lignes qui vont produire du texte dans le fichier de configuration.

En particulier, les lignes précédant la commande `#openfile` peuvent définir les macros `DotG_OUTPUTFILE` qui va identifier le nom du fichier de configuration qui devra être créé (voir la section 3).

Il est possible de trouver plusieurs paires de commandes `#openfile/#closefile` dans un fichier DOTG, mais elles ne doivent pas être imbriquées.

Les prédicats et la commande **#predicate**

La commande `#predicate` définit des macros prédicats. Ce sont des macros particulières, à valeur vraie ou fausse. Une telle définition repose sur la comparaison de la valeur d'une *commande* avec des *motifs*. Une *commande* est une commande Unix quelconque. Un *motif* est une expression régulière de noms de fichiers.

Par exemple, le prédicat `FIL` :

```
#predicate FIL hostname *.fil.univ-lille1.fr
```

prendra la valeur vraie si et seulement si le résultat produit par la commande `hostname` concorde avec l'expression régulière `*.fil.univ-lille1.fr`.

Les commandes `#if`, `#ifnot`, et `#endif`

Ces commandes permettent de traiter ou non un bloc de lignes DOTG selon qu'un prédicat est défini à vrai ou non. Par exemple,

```
#ifnot FIL
# L'imprimante qui jouxte mon bureau
setenv PRINTER didot
#endif
#if FIL
# Utilisation de l'imprimante par defaut
unsetenv PRINTER
#endif
```

On peut trouver plusieurs prédicats associés à un `#if`. Le bloc est traité si l'un des prédicats est vrai.

Les constructions `#if/#endif` peuvent être imbriquées.

Le `#endif` correspondant à un `#if` doit être dans le même fichier source DOTG.

La commande `#info`

La commande `#info` indique à `dotg` de générer quelques lignes indiquant que c'est elle qui à produit le fichier, par exemple sur le modèle suivant :

```
### Do not edit, this is an AUTOMATICALLY GENERATED file
### Generated by dotg for phm from dot-login.csh
```

Ces lignes doivent être des lignes de commentaire pour le fichier de configuration généré. La macro spéciale `Dotg_COMMENTCHAR` configure cela.

3 Macros spéciales

Un certain nombre de macros spéciales sont reconnues par `dotg`. Elles permettent de configurer le comportement de la commande.

Nom du fichier généré

La macro `DotG_OUTPUTFILE` a pour valeur le nom du fichier de configuration devant être généré. La valeur par défaut est – désignant la sortie standard.

Mode des fichiers générés

La macro `DotG_OUTPUTMODE` a pour valeur le mode (sous forme numérique telle `440` ou sous forme symbolique telle `ug=r`) qui sera celui des fichiers générés. Le mode par défaut est `444`, en effet les fichiers générés n'ont pas à être édités à la main par l'utilisateur.

Sauvegarde des fichiers générés

L'ancien contenu d'un fichier généré est sauvegardé dans un fichier dont le suffixe est contenu dans la macro `Dotg_OLDSUFFIX`, par défaut `.old`.

Génération de commentaires

La valeur de la macro `Dotg_COMMENTCHAR` est utilisé comme préfixe des lignes de commentaires générées par `dotg`, par exemple par la commande `#info`.

4 Syntaxe et options de la commande dotg

La syntaxe de la commande `dotg` est la suivante :

```
dotg [options] dotg-file...
```

ainsi on peut traiter un ou plusieurs fichiers DOTG.

Les options sont les suivantes :

-D Permet de définir des macros comme ceci :

```
-DOUTPUTDIR=/users/phm
```

Ce qui est équivalent à un `#define` dans le fichier.

-T Permet de définir un prédicat à vrai comme ceci :

```
-TFIL
```

-F Permet de définir un prédicat à faux.

--verbose `dotg` indique l'état d'avancement des traitements au cours de son exécution.

--silent Rend l'exécution silencieuse.

--help Affiche un message d'usage de la commande sur la sortie standard et termine sur un succès.

--version Affiche les informations de version sur la sortie standard et termine sur un succès.

5 Exemple

L'exemple introductif suivant illustre l'utilisation de `dotg`. Je désire générer le fichier de configuration de mon shell de connexion (fichier `$HOME/.login`) pour deux configurations : ma machine personnelle et mon compte sur le réseau enseignement au FIL.

Dans le fichier `common.dotg` je définis différents prédicats :

```
;;;-----------------------------
;;; common.dotg
;;; definitions de prédcicats et macros DOTG
;;; fichier inclus par mes differents fichiers DOTG
;;;------------------------------------------------------------

#predicate MAISON hostname *.homelinux.net
#predicate LIFL hostanme *.lifl.fr
#predicate FIL hostname *.fil.univ-lille1.fr
#predicate PORTABLE hostname baroudeur.*

#predicate LINUX uname Linux
#predicate CYGWIN uname CYGWIN*

#if PORTABLE MAISON
#define OUTPUTDIR /duchmol
#endif
#if FIL
#define OUTPUTDIR /home/licence/duchmol
#endif
```

```
#define Dotg_OLDSUFFIX ~
```

Ce fichier suppose le comportement suivant des commandes `hostname` et `uname` :

– sur ma machine personnelle :

```
% hostname
duchmol.homelinux.net
% uname
Linux
```

– sur une machine du réseau du FIL :

```
% hostname
lxo4.fil.univ-lille1.fr
% uname
Linux
```

Dans mon fichier `.login` je désire :

– définir mon éditeur préféré en positionnant la variable `EDITOR;`
– définir mon imprimante préférée en positionnant ou non la variable `PRINTER;`
– utiliser le fichier de configuration fourni par les administrateurs sur le réseau du FIL.

Je n'ai qu'un éditeur préféré, mais l'imprimante que j'utilise à la maison est nommée `didot`[1], alors que le système des salles TP est prévu pour fonctionner sans définition de `PRINTER`. Le fichier de configuration pour Linux fourni par les administrateurs est `/home/admin/var/envi/envi.std.linux`.

Soit le fichier `dot-login.csh` suivant

```
###------------------------------
### dot-login.csh
### source dotg de mon .login
###---------------------------------------------------------

#include common.dotg

#define DotG_OUTPUTFILE <<OUTPUTDIR>>/.login
#define DotG_OUTPUTMODE 444
#define DotG_COMMENTCHAR #

#openfile
###------------------------------
### My dot login file
###---------------------------------------------------------
#info

# Mon editeur : emacs ....
setenv EDITOR emacs

#if MAISON
# L'imprimante qui jouxte mon bureau
setenv PRINTER didot
#endif
#if FIL
# Utilisation de l'imprimante par defaut
unsetenv PRINTER
#endif

#if FIL
#if LINUX
```

[1]Les DIDOT sont une célèbre famille d'imprimeurs des XVIII^{e} et XIX^{e}.

```
# Environnement du FIL
source /home/admin/var/envi/envi.std.linux
#endif
#endif
```

La commande `dotg` générera dans mon fichier `.login`

```
###-----------------------------
### My dot login file
###-----------------------------------------------------------
### Do not edit, this is an AUTOMATICALLY GENERATED file
### Generated by dotg for duchmol from dot-login.csh

# Mon editeur : emacs ....
setenv EDITOR emacs

# L'imprimante qui jouxte mon bureau
setenv PRINTER didot
```

à la maison, et

```
###-----------------------------
### My dot login file
###-----------------------------------------------------------
### Do not edit, this is an AUTOMATICALLY GENERATED file
### Generated by dotg for duchmol from dot-login.csh

# Mon editeur : emacs ....
setenv EDITOR emacs

# Utilisation de l'imprimante par defaut
unsetenv PRINTER

# Environnement du FIL
source /home/admin/var/envi/envi.std.linux
```

en salle TP.

6 Votre projet

Développements progressifs

Il s'agit de développer une version de base de la commande `dotg` pour ensuite y intégrer les fonctionnalités supplémentaires.

On peut distinguer les étapes suivantes :

- Gestion des fichiers inclus.
- Gestion des macros.
- Gestion de la production conditionnelle de lignes.
- Gestion des arguments de la commande.

À chacune de ces étapes, il sera nécessaire de valider votre commande par des essais bien choisis.

Rendre votre projet

Lors d'une démonstration de votre logiciel, vous devrez rendre :

- le source Bourne shell de votre commande `dotg` ;
- un jeu de fichiers de configuration, tests de votre commande ;
- un brève documentation de votre commande. Cette documentation comporte :
  - une description des fonctionnalités de votre commande destinée à un utilisateur,

– une description de votre implantation de la commande `dotg`. Elle est destinée à un développeur qui désirerait étendre ou corriger votre implantation.

Cette documentation est à rendre sous forme de fichiers textes ou de fichiers HTML *simples*.