USTL
Licence d'informatique
Module de C/Unix

Projet Unix

m2h — Conversion de manuels Unix en HTML

Philippe MARQUET

Octobre 2000

Nous allons développer une commande `m2h` qui va transformer les pages du manuel Unix en pages HTML qui pourront être consultées depuis un navigateur/butineur Web.

1 Organisation des pages de manuel Unix

Les pages du manuel Unix sont disponibles en ligne sur toute installation du système. On utilise habituellement la commande `man` pour afficher ces pages. Les pages de manuel sont organisées en sections. Les commandes courantes sont par exemple dans la section 1. Ainsi on référence la commande `man` en précisant sa section par `man(1)`. On invoque alors la commande

```
% man 1 man
```

pour obtenir la page de manuel.

Les sources des pages de manuel sont au format `nroff/troff`. Elles sont organisées en une hiérarchie, à raison d'un répertoire par section du manuel. Par exemple, les pages de la section 1 sont dans le répertoire `man1` et la page de la commande `ls(1)` est dans le fichier `man1/ls.1`.

Il est possible que ces pages soient compressées. On peut ainsi trouver un fichier `man1/ls.1.gz` (compression par `gzip(1)`) ou un fichier `man1/ls.1.bz2` (compression par `bzip2(1)`).

Les pages de manuel se trouvent habituellement dans le répertoire `/usr/man`. La variable d'environnement `$MANPATH` contient une liste des répertoires contenant les pages de manuels. Nous pouvons ignorer ces détails et nous reposer sur l'invocation de la commande `man(1)` pour retrouver une page de manuel. La combinaison des options `-c` et `-W` de `man` affiche le nom du fichier source de la page de manuel demandée. Par exemple :

```
% man -W -c 1 ls
/usr/man/man1/ls.1.bz2
```

2 Organisation du source d'une page du manuel

Les fichiers sources des pages de manuel sont au format `nroff/troff` et utilisent les macros du paquetage **man**. Sur une machine Linux, la page de manuel `man(7)` donne une description de ce paquetage.

En bref, un fichier `troff` est composée de lignes de texte entrecoupées de lignes de formatage du texte. Ces lignes de formatage sont constituées d'un nom de macro qui débute forcément par un point et d'éventuels paramètres de la macro. La macro `."` introduit un commentaire. L'exemple suivant est extrait de la page de manuel de `ls(1)` :

```
." LS fileutils manual page rel. 7.65
.TH LS 1 "December 1999" "GNU fileutils 4.0" "FSF"
.SH NAME
ls \- list directory contents
```

Id : m2h.tex,v 1.7 2000/11/28 18 :34 :54 marquet Exp

```
.SH SYNOPSIS
.B ls
[
.I OPTION
]... [
.I FILE
]...
.PP
List information about the FILEs (the current directory by
default).
```

La macro `.TH` introduit le titre, et les haut et bas de page de la page de manuel. Remarquez que les arguments de la macro sont séparés par des espaces ; aussi on utilise les guillemets `"` pour coder un espace au sein d'un argument. La macro `.SH` introduit une section du manuel. Les macros `.B` et `.I` indiquent respectivement d'utiliser une fonte grasse et italique. La macro `.PP` débute un nouveau paragraphe.

Un affichage de la page pourra rendre quelque chose comme :

LS(1)	FSF	LS(1)

NAME
ls - list directory contents

SYNOPSIS
ls [*OPTION*]... [*FILE*]...
List information about the FILEs (the current directory by default).

On peut aussi trouver des séquences d'échappement au sein de source `troff`. Ces séquences commencent par un "\". La séquence "`\e`" code le caractère "\" ; la séquence "`\-`" le tiret "–". Les séquences "`\f`*x*" changent la fonte courante. "`\f1`" ou "`\fR`" désigne la fonte Roman (celle par défaut) ; "`\f2`" ou "`\fI`" une fonte italique ; "`\f3`" ou "`\fB`" une fonte grasse. Ainsi les deux extraits suivants de code `troff` sont équivalents :

```
SYNOPSIS
.B ls
[
.I OPTION
]... [
.I FILE
]...
```

et

```
SYNOPSIS \fBls\fR [\fIOPTION\fR ]... [\fIFILE\fR]...
```

Parmi les sections d'une page de manuel, on peut trouver une section *SEE ALSO* qui renvoie à des commandes ¡¡ proches ¿¿ de la commande courante. Voici un extrait du source de la page de manuel de la commande `man(1)` :

```
.SH "SEE ALSO"
apropos(1), whatis(1), less(1), groff(1).
```

3 La commande `m2h`

La syntaxe de la commande `m2h` est la suivante :

`m2h` [`-fhilostx`] [*`manpage`*|*`directory`*|[*`section`*]*`command`*]...

La commande produit une page HTML par page de manuel. Si on précise un fichier c'est ce fichier qui est traité. Si on précise un répertoire, c'est l'ensemble des pages de manuel de ce répertoire qui sont traitées. Si on précise une commande, c'est la page de manuel de cette commande qui est traitée. On peut alors préciser une section du manuel dans laquelle rechercher la page.

Les options de `m2h` sont les suivantes :

-f Précise de produire la page demandée et les pages suivantes, ce de manière récursive. Les pages suivantes sont celles qui sont référencées dans la section *SEE ALSO* du manuel.

-h Précise de produire des liens hypertextes vers les pages des commandes référencées dans la section *SEE ALSO*.

-i Produit un index au sein de la page HTML générée. Cet index est composée d'une liste de liens hypertextes vers les différentes sections de la page.

-l *logfile* (par défaut la sortie standard). Produit l'état d'avancement de la commande, les problèmes rencontrés (par exemple la liste des macros `troff` inconnues) dans le fichier précisé.

-o *outputdir* (par défaut **.**). Produit les fichiers HTML dans le répertoire *`outputdir`*.

-s *stylefile* utilise le fichier de style (en général `.m2hs`) pour réaliser les conversions des séquences d'échappement `troff` en balises HTML. Vous avez à définir un format le mieux adapté pour ces fichiers de style. Par exemple un tel fichier peut coder les séquences "`\fB`" et "`\fR`" de la manière suivante :

```
\fB.*\fR        <B>\1</B>
```

-t Produit une page au format texte et non au format HTML. Cette option est principalement utilisée pour mettre au point la commande `m2h`.

-x Produit dans le répertoire de sortie (voir option `-o`) une page `index.html` qui référence l'ensemble des pages de manuel générées.

A Le langage HTML

HTML est un langage de composition de documents/pages destinés à être visualisés par un navigateur/butineur. Un ensemble de *balises* déterminent la présentation et l'organisation de la page. Un élément HTML se compose

- de la balise ouvrante (de la forme <tag>) ; cette balise ouvrante peut comporter des attributs et leur valeur ;
- d'un texte ;
- de la balise fermante (de la forme </tag>).

Un élément prend donc l'une des formes suivantes :

```
<tag_name> text </tag_name>
<tag_name attribute_nalme=argument> text </tag_name>
```

Par exemple,

```
<TITLE>Conversion de manuels Unix en HTML</TITLE>
```

indique le titre du document alors que

```
ceci est <em>important</em>
```

précise que cette partie du texte doit être mise en valeur (*emphasis* en anglais) ; elle sera par exemple présentée en italique ou dans une couleur donnée.

En particulier l'élément

```
<a href="URL">text</a>
```

définit un *hyperlien* ou *lien hypertexte* vers un autre document (celui référencé par `URL`).

Un bon moyen d'apprentissage/compréhension du langage HTML est de regarder les sources de ¡¡ bonnes ¿¿ pages que vous trouverez sur le Web. (Les navigateurs proposent tous une fonction ¡¡ afficher le source ¿¿ ou ¡¡ view page source ¿¿.)

De nombreuses informations sur HTML sont disponibles sur le Web :

– Daniel Boivin et Laurent Gauthier de l'université de Laval au Québec proposent leur ¡¡ Manuel illustré de programmation en HTML ¿¿.
  `http://www.grr.ulaval.ca/grrwww/manuelhtml.html`
– Les ¡¡ Academic Computing Services ¿¿ de l'université du Kansas proposent une brève introduction à HTML (en anglais).
  `http://www.cc.ukans.edu/~acs/docs/other/HTML_quick.shtml`
– La NCSA de l'université d'Illinois à Urbana-Champaign propose une autre introduction à HTML (en anglais).
  `http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html`
– Éric Tilton donne quelques conseils de style pour la rédaction de documents HTML (en anglais).
  `http://www.ology.org/tilt/cgh/`

L'ensemble de ces références est accessible ¡¡ d'un simple clic ¿¿ à partir de la version en ligne de ce sujet : `http://www.lifl.fr/~marquet/ens/cu/m2h/`

Un exemple de source HTML pouvant être généré pour l'extrait de la page de manuel de `ls(1)` est le suivant :

```
<HTML>
 <HEAD>
  <TITLE>Manpage of LS</TITLE>
 </HEAD>
 <BODY>
  <H1>LS</H1>
  <P>
  Section: FSF (1)
  <BR>Updated: December 1999
  <HR>

  <H2>NAME</H2>
  <P>
  ls - list directory contents

  <H2>SYNOPSIS</H2>
  <P>
  <B>ls</B>
  [
  <I>OPTION</I>
  ]... [
  <I>FILE</I>
  ]...

  <P>
  List information about the FILEs (the current directory by default).
 </BODY>
</HTML>
```

B Le langage `troff`

On trouvera des informations concernant `troff` sur le Web ; notamment à partir de la page de Richard Stevens : `http://www.kohala.com/start/troff/troff.html`.

La page de manuel `man(7)` décrit le format et les macros utilisées dans les sources `troff` des pages de manuel.

Bien entendu les sources de pages de manuel peuvent (doivent !) être consultées à titre d'exemple de sources `troff`.

C La commande `quote`

La commande `quote` lit sur son entrée standard des lignes de mots. Les mots sont séparés par des espaces. On utilise les guillements (dits anglais) `"` pour inclure des espaces dans des mots. `quote` produit ces mêmes lignes sur sa sortie standard mais assure que tous les mots sont entre quote `'`.

Exemple :

```
% cat foo.txt
foo bar gee
baz "qux corge" grault
garply    waldo "" plugh " thud" ""
% quote < foo.txt
'foo' 'bar' 'gee'
'baz' 'qux corge' 'grault'
'garply'    'waldo' '' 'plugh' ' thud' ''
```

Le code C de la commande `quote` est le suivant :

```
#include <stdio.h>

#define QUOTECHAR '\''

main (void) {

  int c ;
  enum state {OUTWORD, INQUOTE, INWORD} s = OUTWORD ;
  while ((c=getchar()) != EOF) {
    switch (c) {
      case ' ':
        switch (s) {
          case OUTWORD: case INQUOTE:
            putchar(c) ; break ;
          case INWORD:
            putchar(QUOTECHAR) ; putchar(c) ; s=OUTWORD ; break ;
        }
        break ;
      case '"':
        switch (s) {
          case OUTWORD:
            putchar(QUOTECHAR) ; s=INQUOTE ; break ;
          case INQUOTE:
            putchar(QUOTECHAR) ; s=OUTWORD ; break ;
          case INWORD:
            putchar(QUOTECHAR) ; break ;
        }
```

```
        break ;
      case '\n':
        switch (s) {
          case OUTWORD: case INQUOTE:
            putchar(c) ; break ;
          case INWORD:
            putchar(QUOTECHAR) ; putchar(c) ; s=OUTWORD ; break ;
        }
        break ;
      case QUOTECHAR:
        switch (s) {
          case OUTWORD:
            printf ("%s", "'\\'") ; s=INWORD ; break ;
          case INQUOTE:
          case INWORD:
            printf ("%s", "\\'") ; break ;
        }
        break ;
      default:
        if (s == OUTWORD) { s=INWORD ; putchar(QUOTECHAR) ;}
        putchar(c) ;
    }
  }
  if ((s == INQUOTE) || (s == INWORD)) putchar(QUOTECHAR) ;
}
```

Pour construire la commande `quote`, sauver le source C dans un fichier `quote.c` puis executer la commande

```
% make quote
```