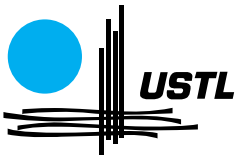

Licence d'informatique
Module de C/Unix

Projet Unix

`ahp` — Another HTML Preprocessor

Philippe MARQUET

Octobre 2001

La commande `ahp` que vous allez développer est inspirée de la commande `gtml` développée initialement par Gihan PERERA et maintenant par Bruno BEAUFILS.

1 Pourquoi `ahp` ?

HTML est un langage relativement puissant de mise en forme de pages Web. Quelques informations et références sur HTML sont données en annexe A de ce document.

Une des limitations majeures d'HTML est la difficulté de maintenir un ensemble de pages de manière cohérente. Les outils tels `gtml`, ou la commande `ahp` que vous allez écrire, ajoutent des fonctionnalités au langage HTML et facilitent par exemple la mise en place d'un « look'n fell » unique pour un ensemble de pages.

De nombreux documents HTML contiennent des informations redondantes supposées être cohérentes pour un ensemble de documents : des barres de navigation, une couleur ou une image de fond, des haut ou bas de page, un logo... Une approche est d'identifier ces informations et de factoriser leur mise à jour par l'utilisation d'un préprocesseur.

Par exemple, une simple macro permet de gérer un bas de page standard. Ce bas de page pourra contenir des icônes de navigations, des renseignements sur l'auteur... En modifiant la définition de cette macro, on pourra changer d'un seul coup la présentation de tout un ensemble de documents HTML. De même, on pourra définir une fois pour toutes des macros pour les URL souvent utilisées.

Le langage AHP est une extension du langage HTML. `ahp` est un préprocesseur de code HTML : il transforme du code AHP en code HTML.

Les fonctionnalités de base de `ahp` sont :

- l'inclusion de fichiers ;
- la définition et l'expansion de macros ;
- la production conditionnelle de texte.

2 Les commandes AHP

Les commandes AHP sont les suivantes :

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

`#define` crée une nouvelle macro qui sera utilisé comme un raccourci.

`#undef` supprime la définition d'une macro.

`#ifdef/#ifndef/#endif` permettent la production conditionnelle de code HTML.

`#openfile/#closefile` marquent le début et la fin de production du code HTML.

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

Parmi les lignes d'un fichier AHP qui ne correspondent pas à des commandes, on distingue :

Id : ahp.tex,v 1.4 2001/11/06 17 :48 :15 marquet Exp

– celles comprises entre un `#openfile` et un `#closefile`. Elles sont lues comme du code HTML et sont produites dans le fichier destination une fois les substitutions de macros réalisées ;
– celles situées hors un bloc `#openfile/#closefile`. Elles sont ignorées.

2.1 La commande `#include`

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

```
#include "common.ahp"
```

`ahp` remplace cette ligne par le contenu du fichier `common.ahp`.

Ce fichier peut contenir d'autres commandes AHP.

Le nom du fichier peut être contenu dans une macro AHP (voir ci-dessous).

2.2 La commande `#define` et l'utilisation des macros

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

```
#define MY_EMAIL phm@lifl.fr
```

définit la macro `MY_EMAIL` dont la valeur est « `phm@lifl.fr` ».

L'utilisation de cette macro, c'est-à-dire la substitution de son nom par sa valeur, est spécifiée dans le texte AHP par l'utilisation de chevrons :

```
<<MY_EMAIL>>
```

D'autres choses sont possibles avec les macros :
– la valeur de macros peut être définie sur la ligne de commande de `ahp` (voir section 3) ;
– il existe un certain nombre de macros prédéfinies (voir section 4) ;
– 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 documents ;
– la valeur d'une macro peut elle même contenir le nom d'une autre macro, par exemple :

```
#define MAILTO_ME <a href="mailto:<<MY_EMAIL>>">
```

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 `MY_EMAIL` après la définition de la macro `MAILTO_ME`.
– l'utilisation d'une macro non définie provoque une erreur.

La commande `#undef` supprime la définition d'une macro :

```
#undef MY_EMAIL
```

Si la macro n'existe pas préalablement, la commande ne fait rien ou produit un avertissement (warning).

2.3 Les commandes `#openfile` et `#closefile`

Un couple de commande `#openfile/#closefile` détermine un bloc de lignes qui vont produire un fichier HTML.

En particulier, les lignes précédant la commande `#openfile` peuvent définir les macros `OUTPUT_DIR` et `EXTENSION` qui vont identifier le nom du fichier HTML qui devra être créé (voir les sections 4.3 et 4.4).

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

2.4 Les commandes `#ifdef`, `#ifndef`, et `#endif`

Ces commandes permettent de traiter ou non un bloc de lignes AHP selon qu'une macro AHP est définie ou non. Par exemple,

```
#ifdef FIL_URL
<A HREF="<<FIL_URL>>">Formations en informatique de Lille</A>
#endif
#ifndef FIL_URL
Formations en informatique de Lille
#endif
```

Les constructions `#if*def/#endif` peuvent être imbriquées.
Le `#endif` correspondant à un `#if*def` doit être dans le même fichier source AHP.

3 Syntaxe et options de la commande `ahp`

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

```
ahp [options] ahp-file...
```

ainsi on peut traiter un ou plusieurs fichiers AHP.
Les options sont les suivantes :

`-D` Permet de définir des macros comme ceci :

```
-DMY_EMAIL=phm@lifl.fr
```

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

`--verbose` `ahp` 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.

L'exécution de `ahp` génère un fichier `.html`[1] pour chacun des fichiers AHP traités.

4 Macros spéciales

4.1 Nom du fichier généré

La macro `ROOT_PATH` a pour valeur le chemin relatif du répertoire courant au répertoire racine, c'est-à-dire au répertoire dans lequel réside le fichier AHP.
La macro `FILENAME` a pour valeur le nom du fichier HTML produit, sans extension et sans information de répertoire.

4.2 Chemin de recherche pour les fichiers inclus

`ahp` recherche les fichiers inclus dans le répertoire contenant le fichier traité puis dans le répertoire courant (c'est-à-dire le répertoire dans lequel la commande `ahp` a été lancée).
La macro `INCLUDE_PATH` peut être définie. `ahp` interprète cette macro comme une liste de répertoires (séparés par des deux-points `:`) dans lesquels les fichiers inclus sont recherchés. Ces répertoires peuvent être absolus ou relatifs au fichier AHP.

[1] Cette extension peut être changée, voir la macro `EXTENSION`, section 4.

4.3 Répertoire de production

Par défaut, `ahp` produit ses fichiers résultats dans le même répertoire que les fichiers sources. Il est possible de préciser le nom d'un répertoire dans lequel produire les résultats en définissant la macro `OUTPUT_DIR`.

Le nom relatif du fichier produit par rapport au répertoire `OUTPUT_DIR` sera identique au nom relatif du fichier AHP par rapport au répertoire courant. Il est ainsi possible de créer une hiérarchie de fichiers HTML identique à une hiérarchie de fichiers AHP.

4.4 Suffixe des fichiers résultats

Par défaut, les fichiers créés portent l'extension `.html` au lieu de l'extension `.ahp` des fichiers sources. La valeur de la macro `EXTENSION` permet de choisir l'extension que porteront les fichiers produits.

5 Extensions de la commande `ahp`

Les sections précédentes présentent une version minimale de la commande `ahp`. Nous définissons ici les extensions que vous pourrez intégrer dans une seconde version de vos développements. Dans un premier temps, ne vous arrêtez pas sur ces extensions.

5.1 La commande `#define+`

La commande `#define+` ajoute une valeur à la définition d'une macro. Par exemple, suite aux commandes

```
#define  WORDS foo bar
#define+ WORDS gee
```

La valeur de la macro `WORDS` sera « `foo bar gee` ».

5.2 La commande `#if`

La commande `#if` permet de traiter un blocs de lignes AHP en fonction de la valeur d'un prédicat. Les prédicats permettent e tester l'égalité (opérateur ==) ou la non-égalité (opérateur !=) de deux valeurs. Ces valeurs peuvent comporter des macros AHP, par exemple :

```
#if <<FOOBAR>> == foo<<BAR>>
```

Bine entendu une commande `#endif` clos le bloc de lignes controlé par la commande `#if`.

5.3 La commande `#else`

Sans surprise, la commande `#else` identifie le début d'un bloc de lignes qui ne seront traitées par AHP que si la commande `#if*` précédente n'était pas un succès.

5.4 La commande `#system`

La commande `#system` produit dans le fichier HTML le résultat d'une commande shell. Par exemple le code AHP

```
#system echo $USER
```

affiche le nom d'utilisateur Unix.

5.5 L'option `-C`

Comme l'option `-D`, l'option `-C` définit une macro. Cependant, les redéfinitions de cette macro dans les fichiers AHP seront sans effet. Il est ainsi possible de surcharger sur la ligne de commande les définitions données dans les fichiers AHP.

5.6 L'option `--makedep`

L'option `--makedep` de la commande `ahp` ne génère pas les fichiers HTML mais produit sur la sortie standard les dépendances entre les fichiers HTML devant être produits et les fichiers AHP sources en prenant en compte les commandes AHP `#include`. Cette option est l'équivalent pour `ahp` des options `-M` ou `-MM` du compilateur C `gcc`. Elle aidera à la production automatique de fichier Makefile pour les fichiers AHP.

6 Rendre votre projet

Vous devez rendre :
- le source Bourne shell de votre commande `ahp` ;
- un jeu de fichiers AHP, 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 `ahp`. Elle est destinée à un développeur qui désirerait étendre ou corriger votre implantation.

Il est évidement bienvenu de rendre cette documentation sous forme de fichier AHP pouvant être par traités par votre commande `ahp`, par exemple à l'aide d'un Makefile.

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

Par exemple,

```
<TITLE>ahp --- Préproceseur 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* 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 (les références données ici sont en anglais) :

- Le « World Wide Web Consortium » (W3C) propose un document de Dave Raggett « Getting started with HTML ».
  `http://www.w3.org/MarkUp/Guide/`
- Les « Academic Computing Services » de l'université du Kansas proposent une brève introduction à HTML.
  `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.
  `http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html`
- Éric Tilton donne quelques conseils de style pour la rédaction de documents HTML.
  `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/ahp/`.