Projet CPI 2016 printemps

P. Pittoli

Le projet consiste à faire deux exercices qui vous feront écrire des scripts shell. Chacun de ces scripts devra vérifier les paramètres qui lui sont donnés et quitter avec un numéro de retour pertinent (0 en cas de succès, autre chose en cas d'erreur). Chaque script sera également accompagné d'une page de manuel pour décrire comment l'utiliser (voir les consignes en dernière page).

Le sujet du projet peut être mis à jour, allez vérifier régulièrement la dernière version sur le site t.karchnu.fr/cours/cpi.

1 Le jour où j'ai remplacé Google

Et je l'ai fait en un après-midi. Facile.

Pour ce premier exercice, nous allons faire un moteur de recherche simpliste. L'exercice se découpe en plusieurs scripts shell très simples, qui devront être utilisables séparément. Si quelque chose n'est pas spécifié clairement dans le sujet, faites comme bon vous semble et justifiez vos choix.

1.1 get-clean-domain-name

Ce script va lire sur son entrée standard des URL (une par ligne) et affiche uniquement le nom de domaine. Il faut vérifier les entrées données ! Exemple :

```
$ echo https://fr.wikipedia.org/wiki/Épistémologie | get-clean-domain-name
fr.wikipedia.org
```

1.2 web-get-title

Ce script va lire sur son entrée standard des url et ira chercher le contenu de la balise « <title> » de la page. Il faut vérifier les entrées données ! Exemple :

```
$ echo https://love2d.org/ | web-get-title
LÖVE - Free 2D Game Engine
```

1.3 web-get-links

Ce script prend en paramètre une url et ira chercher tous les liens présents sur la page (balises « <a> »). Il faut vérifier que le paramètre donné est bien une URL. Exemple :

```
$ web-get-links http://framasoft.org
https://framasoft.org
http://framablog.org/2015/10/05/degooglisons-saison-2-ils-ne-savaient-pas-que-cetait-impossible-alors
http://framabook.org/grise-bouille-tome-1/
...
```

1.4 crawler

Ce script est la glu qui lie tous les scripts précédents. Son objectif est de trouver les liens d'une URL donnée, les parcourir, et récupérer le titre de chaque page. Le script comportera une option « -n » pour donner une limite à la recherche de liens, cela donnera la profondeur de la recherche. La sortie doit ressembler à « URL nom de domaine titre » comme ceci :

```
$ crawler -n 2 https://en.wikibooks.org/wiki/LaTeX
https://en.wikibooks.org/wiki/LaTeX en.wikibooks.org LaTeX - Wikibooks, open books
https://www.mediawiki.org/wiki/Help:Contents www.mediawiki.org Help:Contents - MediaWiki
...
```

Attention à ne pas parcourir plusieurs fois le même lien. Il serait intéressant aussi de suivre les liens relatifs.

Vous êtes libre également d'aller plus loin que ce qui est explicitement demandé. Peut-être qu'au bout d'une certaine taille de fichier (ou de nombre de lignes) on souhaiterait archiver (et compresser) notre base de données et écrire dans un autre fichier ? Les sites web sont-ils accessibles en IPv6 ? Est-ce qu'ils respectent les normes du w3c ? Quelle est le poids de leur page ? Etc.

Vous pouvez aussi empaqueter votre programme pour qu'on puisse l'installer depuis debian (.deb). Des points bonus à la clé !

2 La musique c'est bien !

Quand elle est partagée c'est mieux.

Jamendo.com, très beau site, de la musique gratuite et souvent libre. On peut y écouter de la musique sans limite, y découvrir des groupes sympas, et surtout on peut télécharger des albums pour les écouter à la maison. Problème, ces albums arrivent dans des fichiers .zip, avec un nom pas très propre.

Votre mission (que vous acceptez) sera de faire un script qui prend un répertoire en paramètre (là où vous faites vos téléchargements), et qui pour tous les albums que vous avez téléchargés vous crée un répertoire avec un nom correct et y dépose vos musiques.

Exemple, vous recevez le fichier « For The Broken - From Sinners To Sinners - 139563 — Jamendo - MP3.zip », vous allez créer le répertoire « For The Broken - From Sinners To Sinners ». Pour être plus explicite, le nom original contient « - 139563 — Jamendo - MP3.zip » en trop.

Un second problème est que les fichiers reçus ont parfois le droit d'exécution, ce qui n'est pas normal pour un fichier de musique. De plus, vous souhaitez que seuls vous et votre groupe puissiez lire votre musique, les autres n'ont pas le droit ne serait-ce que de voir les musiques que vous avez téléchargées (vos goûts musicaux ne regardent que vous !). Si vous voyez un autre changement de droit pertinent à faire, dites-le et faites-le.

N'hésitez pas à aller rechercher des exemples sur le site, comme LukHash, Aygan, Binärpilot... autant d'exemples qu'il vous faudra pour « tester votre script » !

Si vous souhaitez aller plus loin que ce qui est demandé, n'hésitez pas ! Des points bonus à la clé ! ;)

3 Consignes supplémentaires

3.1 Écrire une page de manuel

Les pages de manuel de vos scripts comporteront au minimum :
— le nom du script avec une petite description de son utilité
— le nom de l'auteur de la page de manuel
— la date de création de la page de manuel
— une description claire des différentes options possibles
— le synopsis pour savoir comment utiliser le script
— les différents numéros de retour
— les erreurs et limitations de vos scripts
— et tout autre élément qui vous semblera pertinent

Faites `man 7 mdoc` pour apprendre à créer des pages de manuel, je vous conseille également de prendre exemple sur les pages de manuel déjà présentes sur votre système (dans `/usr/share/man/man1/` par exemple). Le nom de votre page de manuel sera le nom de votre script suffixé par « .1 », le numéro « 1 » correspondant à la section dédiée aux manuels d'utilisation d'un programme (`man man` pour vous rafraîchir la mémoire).

3.2 La lisibilité

Le code doit être lisible et commenté. Un code non indenté vaut 0. Un code non commenté vaut 0. Les commentaires doivent être pertinents, et montrer que vous comprenez ce que vous faites. Un saut de ligne de temps en temps ne peut pas faire de mal.

Vous êtes libre d'organiser votre code comme bon vous semble, s'il vous paraît judicieux de séparer le code en plusieurs fichiers pour améliorer la lisibilité ou pour rendre votre code plus modulaire, n'hésitez pas !

Vous avez explicitement le droit de rendre vos sorties d'exécution plus jolies si vous le souhaitez, tant que cela ne nuit pas à la lisibilité du code et que c'est pertinent ; par exemple en écrivant le nom du répertoire que vous allez créer dans l'exercice 2.

3.3 Un tout petit projet INDIVIDUEL

Un code qui ressemble d'un peu trop près à un autre vaut 0. Le projet est à faire individuellement.

3.4 Trouver de l'aide

Les pages de manuel peuvent vous aider, de même que de nombreux sites. N'utilisez un moteur de recherche que si vous ne trouvez pas dans le manuel, vous en apprendrez plus ! ;)

Si vous avez des questions, n'hésitez pas à envoyer un mail ! `p.pittoli CHEZ unistra.fr`

3.5 Le rendu

Le rendu doit se faire dans une archive au format `nom-prenom_projet-cpi.tar.gz`. Le barème sera décidé au doigt mouillé, inutile de me spam pour ça. Bon courage.