DM - ASR7 Programmation Concurrente

Rappels Système

Septembre

I Retour sur les scripts shell

— On considère le script `convertoMP3.sh` suivant. Que fait-il ?

```
#!/bin/bash

# see http://en.linuxreviews.org/HOWTO_Convert_audio_files

#
# Usage: convertomp3 fileextention
#
if [ "x$1" = "x" ];then
  echo 'Please give a audio file extention as argument.'
  exit 1
fi

for i in *.$1
do
  if [ -f "$i" ]; then
  rm -f "$i.wav"
  mkfifo "$i.wav"
  mpv \
   -quiet \
   -vo null \
   -vc dummy \
   -af volume=0,resample=44100:0:1 \
   -ao pcm:waveheader:file="$i.wav" "$i" &
  dest=`echo "$i"|sed -e "s/$1$/mp3/"`
  lame -V0 -h -b 160 --vbr-new "$i.wav" "$dest"
  rm -f "$i.wav"
fi
done
```

— Pourquoi

`$> convertoMP3 webm`

m'indique `command not found` ?

Si vous ne savez pas, retour sur ce qu'est la variable PATH (voire toutes les autres variables d'environnement, ce qu'elles sont et leur utilisation).

— J'ai copié le script dans le fichier `convertoMP3`. Pourquoi la commande
`$>./convertoMP3 m4a`
ne s'exécute pas ?

Revoir les droits des répertoires et des fichiers : notamment ceux des binaires et scripts, qui se doivent d'être exécutables, et pour un script, il faut qu'il soit aussi lisible (voir l'impact des droits du répertoire sur ceux des fichiers qu'il contient !). Pas besoin ici, mais si vous lisez cette correction, il vaut mieux revoir les droits spéciaux comme +s.

— À quoi sert l'extension `.sh` à la fin du nom du fichier ?

C'est purement décoratif : ça peut aider l'utilisateur à savoir ce qu'est le fichier, mais l'information sur son contenu est obtenu avec la commande `file`

— Est-ce que `root` doit placer ce script dans `/usr/bin/` ?

NON ! Ça le serait s'il s'agissait d'une commande utilisable pour tout le système, c-à-d accessible également aux autres utilisateurs : c'est ce qu'il se passe par exemple lors de l'installation de nouveaux *packages* sur la machine. Mais **chacun peut avoir ses propres scripts** pour lui simplifier la vie. L'utilisateur peut créer un répertoire `~/bin` ou `~/.local/bin` pour cela.

— On place `convertoMP3` dans `~/bin/` mais
`$> convertoMP3 wav`
ne fonctionne pas. Pourquoi ?

Il faut vraiment réviser ce qu'est la variable d'environnement `PATH`, et comment ajouter des informations au `PATH` ; et savoir comment faire pour que cette variable soit définie dans tout nouvel environnement créé (gestion de `~/profile`, `~/bash_profile` et `~/bashrc`).

II Scripter pour automatiser – et se simplifier la vie

Jean n'a pas la data sur son smartphone. Il sait par ailleurs que tout transfer est mauvais pour le climat [1] et qu'il n'a pas besoin d'avoir des vidéos en 4k : elles consomment beaucoup de bande passante, épuisent sa batterie pour rien, et consomment en plus de l'espace de stockage s'il les télécharge. Mais il prépare un voyage en train, et avoir à disposition quelques documentaires disponibles sur `arte.tv` [2], quelques épisodes de série, en plus d'un bon livre serait agréable. Or il sait qu'il peut télécharger des vidéos avec l'outil `youtube-dl`...

1. `https://www.euractiv.fr/section/climat/news/cat-videos-online-porn-branded-an-ecological-disaster/`
2. notamment les épisodes `https://www.arte.tv/fr/videos/RC-015330/propaganda/`

II.1 Préparatifs

— Quelle est l'option qui permet d'afficher l'ensemble des flux vidéos et audio de l'URL entrée ?

Utilisation de `man`, `-F` ou `--list-formats`

— Quelles sont les options qui permettent de ne télécharger que le flux audio, de préciser le type de flux audio ogg vorbis, en choisissant la meilleure qualité audio disponible ?

Utilisation de `man`, `-x --audio-format vorbis --audio-quality 0`

— Jean appréciant particulièrement avoir des flux audio avec lui, il souhaite créer un raccourci commande pour cette commande : comment fait-il ?

Il utilise les alias ! Par exemple `alias yda="youtube-dl -x -audio-format vorbis -audio-quality 0"`
placé dans son `~/.bashrc`

II.2 Script

— Jean est aussi fan de XKCD. Cette fois-ci, plus de vidéo youtube, mais Jean aimerait télécharger les images en ligne de commande, pour pouvoir scripter les choses plus facilement. Il sait que l'URL de l'image se trouve derrière la chaîne de caractère `Image URL` sur les pages `https://xkcd.com/`< `numero` >. 1) Comment télécharger la page web (HTML) ? La commande `wget` devrait aider, mais comment a-t-on la liste des options disponibles ?
2) Comment avoir la ligne de la page web qui contient l'URL de l'image ?

`wget URL` pour télécharger n'importe quoi sur le net. Comme à peu près toute commande Unix, on accède à la documentation avec `man wget` dans un terminal (ou on cherche un tuto via son moteur de recherche favori. À noter les options `-r` (récursif) et `--no-parent` pour aspirer une partie ou un site web complet, mais elles ne nous serviront pas ici.

Pour obtenir la ligne, il suffit de faire un `grep`. On peut tenter :
`wget URL | grep "Image URL"`
Ça ne marche pas, car alors, `grep` s'applique à la sortie standard de `wget`, qui est vide, alors que le contenu de la page est envoyé dans un fichier. Seconde tentative :

```
prompt$ wget https://xkcd.com/2418/ --quiet -O - | grep 'Image URL'!
Image URL (for hotlinking/embedding): https://imgs.xkcd.com/comics/metacarcinization.
```

Bingo, ça a marché !

— Jean dispose maintenant de la ligne qui contient l'URL, mais elle est de la forme suivante : `Image URL (for hotlinking/embedding): https://imgs.xkcd.com/comics/metacarcinization.png`
Il faut donc découper la ligne pour en extraire l'URL de l'image. Comment faire ?

Une commande classique est `sed`, qui permet de faire des substitutions d'expressions régulières avec la syntaxe `sed 's/expression-a-remplacer/remplacement/'`. La commande est coupée en 3 lignes pour qu'elle rentre dans la largeur de la page (on a le droit d'entrer un retour charriot derrière un `|` ou un `\`), mais vous pouvez aussi l'entrer sur une seule ligne, ça marche aussi :

```
prompt$ wget https://xkcd.com/2418/ --quiet -O - |
    grep 'Image URL' |
    sed 's/Image URL (for hotlinking\/embedding): //'
https://imgs.xkcd.com/comics/metacarcinization.png
```

On peut maintenant utiliser `wget` pour télécharger cette image :

```
prompt$ wget https://xkcd.com/2418/ --quiet -O - |
    grep 'Image URL' |
    sed 's/Image URL (for hotlinking\/embedding): //' |
    xargs wget -O 2418.png
--2021-01-29 19:22:41--  https://imgs.xkcd.com/comics/metacarcinization.png
Resolving imgs.xkcd.com (imgs.xkcd.com)... 151.101.120.67, 2a04:4e42:1d::67
Connecting to imgs.xkcd.com (imgs.xkcd.com)|151.101.120.67|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60776 (59K) [image/png]
Saving to: '2418.png'
100[=====================================>]  59.35K  --.-KB/s    in 0.02s
2021-01-29 19:22:41 (3.44 MB/s) - '2418.png' saved [60776/60776]
```

On peut pousser un peu plus loin et télécharger plusieurs images d'affilée avec une boucle `while` :

```
for i in $(seq 2400 2418); do
    wget https://xkcd.com/$i/ --quiet -O - |
        grep 'Image URL' |
        sed 's/Image URL (for hotlinking\/embedding): //' |
        xargs wget -O $i.png
done
```

— Script : Jean ne se rappelle plus forcément quel est le dernier numéro qu'il a regardé car il ne regarde pas les anime chaque semaine. Il souhaite : Stocker le numéro du dernier anime regardé dans un fichier getit.param, faire une boucle `while` qui téléchargera la page web du prochain épisode non regardé/téléchargé (l'URL de la page web est toujours du genre `https://xkcd.com/<numero>/`), puis récupérera l'URL de l'image pour la télécharger. La boucle `while` s'arrêtera quand `wget` s'arrêtera sur un status d'erreur parce que la page n'existe pas
Écrivez le script de Jean !