Le script shell `op`

18 mars 2019 — Jawher Jarrey, Hiba Ouni, Pierre Rousselin et Xavier Monnin

Le but de ce sujet est d'écrire petit à petit un script nommé `op` qui pourrait avoir l'aide suivante :

```
op - ouvre un ou plusieurs fichiers

SYNOPSIS
    op [FICHIER] ...
    op --help
    op --version
```

Par exemple, la commande `op fichier.pdf` devrait (si le fichier existe et est lisible) ouvrir ce fichier avec un lecteur de fichiers `pdf` (par exemple `evince`) en tâche de fond. Un fichier portant l'extension `.c` ou `.h` ou les noms `Makefile` ou `makefile` serait ouvert avec votre éditeur favori, etc.

1. On commence par la version la plus simple : créer le script `op` de manière à ouvrir en arrière-plan, avec les applications de votre choix un seul fichier reconnaissable par son extension ou son nom, avec l'application convenable de votre choix. Un exemple d'applications disponibles (mais elles sont nombreuses et bien sûr *vous avez le choix*) :

- pour les images : eye of gnome (`eog`) ;
- pour les documents `pdf` : `evince` ;
- pour les musiques, `cvlc` ;
- et bien sûr votre éditeur de texte favori pour les sources C, les makefiles, ...

2. Tester votre commande, puis lorsqu'elle marche, l'installer dans votre répertoire `~/bin` en vérifiant que ce répertoire est bien dans votre variable `PATH` (modifier cette variable dans `~/.bashrc` si ce n'est pas le cas).

3. Modifier votre script pour qu'il gère convenablement le cas d'un fichier :

a) pour lequel vous n'avez pas de permission de lecture ;

b) qui n'existe pas ;

c) qui est un répertoire (dans ce cas, ouvrir un navigateur de fichiers comme `nautilus`, par exemple) ;

d) qui n'est ni un fichier normal, ni un répertoire.

4. Modifier votre script de façon à ce que l'utilisateur puisser entrer plusieurs fichiers, par exemple :

```
op fichier1.pdf fichier2.txt fichier3.c fichier4.pdf
```

5. Modifier votre script de façon à informer l'utilisateur lorsque le fichier n'a pas pu être ouvert à cause d'un type de fichier non pris en charge.

6. Tester dans le terminal la commande `file` sur certains types différents de fichier. Puis modifier votre script de façon à ce que, si l'extension n'est pas présente ou non reconnue, la sortie de `file` soit utilisée pour, *si possible*, reconnaître le type de fichier (et donc utiliser la bonne application pour l'ouvrir).

7. Prendre en compte :

a) l'option `-h` ou `--help` pour afficher une aide, puis sortir (avec un code de sortie correspondant à un succès). Pour ce faire, on utilisera *une fonction shell* `help()` et un document en-place (*here-document*) de la façon suivante :

```
cat <<FIN_AIDE
bla bla
bla bla
bla bla
bla bla
FIN_AIDE
```

b) l'option `-v` ou `--version` pour afficher des informations de version sur votre programme ;

c) les options non prises en compte (une autre option que les précédentes) en utilisant une fonction shell `usage()`.

8. (Si vous avez le temps...) Faire en sorte que les applications choisies pour chaque type de fichier soient écrites dans un fichier de configuration (par exemple `~/.oprc`) qui sera lu par votre script pour choisir l'application à utiliser. Le fichier de configuration pourrait avoir *par exemple* la forme suivante :

```
text C script makefile -- gedit
tex -- texmaker
jpg png gif -- eog
pdf ps -- evince
mp3 ogg -- cvlc
html, htm -- firefox
```

9. Plutôt qu'une seule application dans ce fichier on pourrait en avoir une liste pour avoir des solutions si l'application préférée n'est pas installée, *par exemple* si la ligne du fichier de configuration est

```
text C script makefile -- gedit, emacs, kate
```

alors `gedit` sera utilisé par défaut, mais s'il n'est pas installé, c'est `emacs` qui sera utilisé et en dernier recours, `kate`.