Pourquoi utiliser la ligne de commande?

Dans un environnement technologique où presque tout a une durée de vie utile de quelques années tout au plus, il est parfois important de se questionner sur les approches les plus durables. Dans le domaine du génie logiciel, seules quelques technologies passent à travers les décennies sans être profondément transformées. Le courriel, le protocole Internet et la culture des systèmes Unix. À travers cette présentation, je veux vous faire découvrir pourquoi il est toujours pertinent d'apprendre à utiliser la ligne de commande Unix à travers l'interface utilisateur la plus courante, GNU Bash.

Quelques références

- (Raymond 2003) The Art of Unix Programming: Un livre classique sur la philosophie des systèmes Unix.
- Unix Acronym List: Une liste des nombreux acronymes utilisés dans les systèmes Unix. Un petit traité d'étymologie de la culture Unix.

Standard établi depuis des décennies

GNU Bash est fourni avec tous les systèmes qui suivent les normes POSIX, un standard établi en 1989 et dont Richard Stallman faisait partie du comité initial. Cette norme est apparue 19 ans après la création du premier système Unix par Kenneth Thompson, Dennis Ritchie et Brian Kernighan.

Référence:

- GNU Bash

Énormément de documentation

Les pages "man"

Les pages `man` sont une forme de documentation standardisée et rigoureuse et qui sont distribuée avec la plupart des logiciels avec une interface en ligne de commande. On doit leur forme à Brian Kernighan. Beaucoup de logiciels permettent aussi d'obtenir de l'aide en insérant `-h` après la commande. Certains manuels `man` ont plusieurs milliers de pages.

Référence:

- Practical UNIX Manuals

Une littérature imposante

Plusieurs centaines de livres sur la programmation en ligne de commandes avec Bash sont édités. De plus, d'innombrables tutoriels sont disponibles en ligne. Bash est la *lingua franca* des informaticiens depuis près de 40 ans.

Référence:

- Recherche ''%%Bash%%'' sur Amazon

- Awesome Bash, une collection de matériel en ligne sur Bash

Une très grande communauté

Le site web Stack Overflow recense plus de 100000 questions et réponses sur Bash

Référence:

- ''%%Bash%%'' sur StackOverflow

Maturité des logiciels

Les logiciels en ligne de commande ont une plus grande maturité que les logiciels avec interface graphique. Ils sont souvent plus anciens, et comme ils n'incluent pas la complexité de gérer l'interface graphique, davantage d'efforts sont mis dans le développement, la sécurité et le respect des normes telles que POSIX.

Indépendance entre les distributions GNU/Linux et autres systèmes Unix

Les logiciels utilisés en ligne de commande, ainsi que le langage Bash, sont des outils communs à toutes les distributions GNU/Linux ainsi que les autres systèmes Unix qui suivent les normes POSIX. Ainsi, ce qui est appris sur un système peut être réutilisé tel quel. Une fois qu'on a appris les bases de la programmation avec Bash, on peut se débrouiller un peu partout.

Parce que c'est amusant

```
#!/bin/bash
sudo aptitude moo
sudo apt-get moo
```

```
#!/bin/bash
cowsay "La ligne de commande c'est amusant"
```

Parce que parfois, on n'a pas le choix

Accès à distance avec SSH

Serveur Web

Comment exécuter un programme?

Les programmes en ligne de commande s'exécutent dans un émulateur de terminal, ou shell.

L'interface utilisateur, ou shell

Une fois un émulateur de terminal chargé, on dispose d'un environnement d'exécution dans lequel nous pouvons entrer des commandes Bash ou le nom d'une application suivi d'arguments. Celui-ci inclus plusieurs variables d'environnement, dont une servira à identifier où se trouvent les programmes exécutables. C'est le PATH. Cette variable contient une liste de tous les répertoires, dans l'ordre, dans lequel nous voulons que Bash cherche les applications.

La commande `which` permet d'obtenir le chemin complet du programme qui sera exécuté lorsque son nom est entré à la ligne de commandes.

Par exemple, si on veut savoir quel interpréteur Python est appelé par défaut:

```
#!/bin/bash
which python
```

Shebang

Les symboles `#!` situé au début de la première ligne d'un script permettent d'identifier avec quel interpréteur nous désirons exécuter ce dernier. Par défaut, si nous appelons un programme depuis Bash avec la syntaxe `./programme`, il sera exécuté avec GNU Bash.

GNU Bash sur Linux

Sur Linux, GNU Bash est l'interpréteur de commandes par défaut utilisé dans les émulateurs de terminaux, tels que Konsole ou Gnome Terminal.

Mac OS X et Windows

- Depuis Mac OS X Panther, l'émulateur de terminal par défaut est Bash
- Depuis Windows 10, il est possible d'installer Bash via le sous-système Linux pour Windows.

Cygwin et Msys2

Cygwin et Msys2 permettent d'installer des environnements compatibles POSIX sur un système Windows. Par contre, contrairement à l'intégration présente dans Windows 10, il s'agit ici de logiciels qui ont été recompilés pour la plateforme Windows.

Cygwin est très complet, mais peut être difficile à installer et à utiliser. Msys2 est facile à utiliser et sera familier pour les utilisateurs de Arch Linux car il utilise le système de distribution de paquetages `pacman`

Référence:

- Cygwin
- Msys2

Est-ce que GNU Bash est un vrai langage de programmation?

Oui, GNU Bash est un langage de programmation complet, qui permet de construire des programmes complexes ainsi que des logiciels. Cependant, il est concu pour écrire des scripts et non des applications.

Référence pour le langage GNU Bash:

- Wikibooks: Programmation Bash

Structures de contrôle

Structure conditionnelle

Une structure conditionnelle permet de modifier l'exécution du programme selon une condition.

```
#!/bin/bash
if [condition]
  then
      commandes
  elif [condition]
  then
      commandes
  else
      commandes
  fi
```

Boucle

La boucle `for` permet d'effectuer un même groupe de commande pour toutes les valeurs d'une liste. Par exemple, si on veut effectuer un traitement sur une liste de fichiers

Variables

Une variable permet de garder une valeur produite par une application ou une fonction et de la réutiliser ultérieurement

On définit une variable comme suit:

```
VARIABLE=valeur
```

Ensuite, pour retrouver la valeur de la variable, on met un signe $ devant. Le programme `echo` affiche tout ce qui lui est passé comme argument à l'écran.

```
echo $VARIABLE
```

Si on désire limiter la portée d'une variable, on précède l'assignation du mot-clé `local`. Au contraire, si on désire garder la variable en dehors du programme qui l'a créée, on utilise le mot-clé `export`.

On peut aussi utiliser la sortie d'un programme à l'intérieur d'une chaîne de caractère en utilisant la syntaxe `$(programme)`. Par exemple, si on veut utiliser la date courante pour nommer un fichier d'archive, on peut utiliser le programme suivant:

```
#!/bin/bash
OF=/media/disque-externe/mon-backup-$(date +%Y%m%d).tar.gz
tar -zcf $OF /home/utilisateur/
```

Composition et redirection

Pour passer la sortir d'une application en entrée à une autre application, on utilisera le “pipe”

```
cat fichier | programme1 | programme2
```

Pour rediriger la sortie d'un programme vers un fichier, on utilise le chevron >. Notons que ceci peut fonctionner aussi dans l'autre sens, en lisant un fichier vers l'entrée d'un programme. On peut même utiliser deux chevrons pour lire un fichier, le transformer et écrire le résultat dans un autre fichier.

Quelles sont les différences entre GNU Bash et PowerShell?

Comment travailler avec ...

Des fichiers

Les systèmes de fichiers sont composés de fichiers et de répertoires. Il existe plusieurs utilitaires de base pour manipuler ces éléments. Tout d'abord, on doit être capable de naviguer à travers le système de fichiers.

- `ls` permet de lister le contenu d'un répertoire
- `cd` permet de changer de répertoire
- `mkdir` permet de créer un répertoire
- `touch` permet de créer un fichier
- `rm` permet de supprimer un fichier
- `cp` permet de copier un fichier
- `mv` permet de déplacer un fichier

On doit ensuite pouvoir lire le contenu d'un fichier. Habituellement, il est mieux de ne pas présumer de la taille d'un fichier.L'avantage de tous les utilitaires de manipulation de fichiers texte est qu'ils lisent les fichiers une ligne à la fois sans les charger en mémoire, on peut dont travailler avec des fichiers de plusieurs gigaoctets sans avoir à se soucier de la performance.

- `less` permet d'afficher le contenu d'un fichier un écran à la fois.
- `head` permet d'afficher les premieres lignes
- `tail` permet d'afficher la fin d'un fichier. On peut aussi suivre l'écriture en direct dans un fichier, ce qui peut être très utile pour suivre un journal en temps réel.

Le réseau Internet

Pour vérifier si un ordinateur distant est en ligne, on peut utiliser la commande `ping`. Pour identifier qui est le propriétaire d'un domaine, on effectue une requête `whois`.

```
#!/bin/bash
whois francoispelletier.org
```

Les utilisateurs et permissions

Le système de permission POSIX est composé de trois valeurs en base octale, qui sont la somme des permissions en lecture, en écriture et en exécution pour le propriétaire, le groupe et les autres utilisateurs.

La permission de lecture ® vaut 4, d'écriture (w) vaut 2 et d'exécution (x) vaut 1.

Pour afficher les permissions d'un répertoire, on utilise l'appel suivant: `ls -l`. Pour les modifier, on utilisera la commande `chmod` suivie de la valeur octale des permissions, où `400` est la plus restrictive et `777` la plus permissive.

La compression de fichiers

Les systèmes GNU/Linux incluent souvent de nombreux utilitaires de compression de fichiers et de répertoires, lesquels ont chacun leurs avantages et leurs désavantages. Les deux formats les plus fréquents sont le `tar.gz` et le `tar.bz2`. Pour créer ces fichiers, on utilise l'application `tar` accompagnée de divers paramètres et interrupteurs, suivi du nom du répertoire à compresser.

Par exemple:

```
#!/bin/bash
tar zcf mon_archive.tar.gz mon_repertoire/
```

Dans le cas où on désire compresser un seul fichier, il est préférable d'éviter l'archive `tar` est d'opter plutôt pour la compression directe par un des utilitaires `gzip` ou `bzip2`

Quelques raccourcis utiles

Il y a quelques raccourcis utiles à connaître pour utiliser la ligne de commane de façon conviviale:

- `Ctrl+c` termine la commande courante
- `Ctrl+d` quitte la session courante
- `Ctrl+w` efface le dernier mot

- `Ctrl+u` efface la ligne entière
- `!!` répète la commande précédente

Effectuer une tâche spécifique

Cette section présente plusieurs exemples de logiciels puissants et versatiles qui sont essentiellement utilisés à partir d'une interface en ligne de commandes.

Manipulation d'images et de vidéos

- Imagemagick permet d'effectuer des opérations d'édition, de composition et de conversion d'images. Il peut lire et écrire plus de 200 formats d'images différents.

```
#!/bin/bash
convert eye.gif news.gif \
-append storm.gif tree.gif \
-background skyblue \
+append result.gif
```

- FFmpeg est un convertisseur audio et vidéo très performant, qui peut aussi recevoir de l'audio et du vidéo en flux continu et en effectuer la conversion en temps réel.

```
#!/bin/bash
ffmpeg -i input.avi -r 24 output.avi
```

Références:

- Imagemagick
- FFmpeg

Télécharger des fichiers

L'utilitaire GNU Wget permet d'effectuer des requêtes web avec la méthode GET du protocole HTTP.

```
#!/bin/bash
wget https://linuq.org/_media/tristan_nitot.zip
```

Il permet aussi de télécharger de manière hiérarchique un site web.

```
#!/bin/bash
wget -mpck --user-agent="" -e robots=off --wait 1 www.foo.com
```

L'application youtube-dl permet de télécharger de nombreux contenus multimédia souvent imbriqués dans des pages web

```
#!/bin/bash
```

```
youtube-dl https://www.youtube.com/watch?v=Wafj7Cqw3Pg
```

Référence:

- Archiving a Website With Wget

Synchroniser des fichiers

rsync est un logiciel qui permet de rapidement transférer des fichiers entre deux emplacements de manière incrémentale. Ceci en fait un outil très efficace pour effectuer des sauvegardes.

```
#!/bin/bash
rsync -avAXHS \
--progress \
--exclude={"/dev/*","/proc/*","/sys/*","/tmp/*",\
"/run/*","/mnt/*","/media/*","/lost+found"} \
/ \
/path/to/backup/folder
```

Référence:

- Site officiel de rsync

Pouvoir reproduire une séquence d'actions dans le futur

Script vs. application

Précédemment, plusieurs exemples d'applications ont été présentés. Les applications ont principalement comme caractéristique d'être conçues pour effectuer une tâche précise, elles comprennent des entrées et des sorties, sont génériques et autonomes.

Les scripts sont des recettes conçues pour répondre à un besoin spécifique de l'utilisateur. Il combinera habituellement plusieurs applications, soit de manière séquentielle, soit de manière chaînée ou en utilisant des structures de contrôle, que nous verrons un peu plus loin.

Le langage GNU Bash permet de construire des recettes qui interagissent directement avec les différentes composantes du système d'exploitation et de l'environnement utilisateur.

D'un point de vue mathématique, une application est l'opération qui détermine la valeur d'une fonction pour un argument donné. Une composition est l'application successive de fonctions à la valeur d'une fonction précédente. C'est souvent le rôle joué par un script.

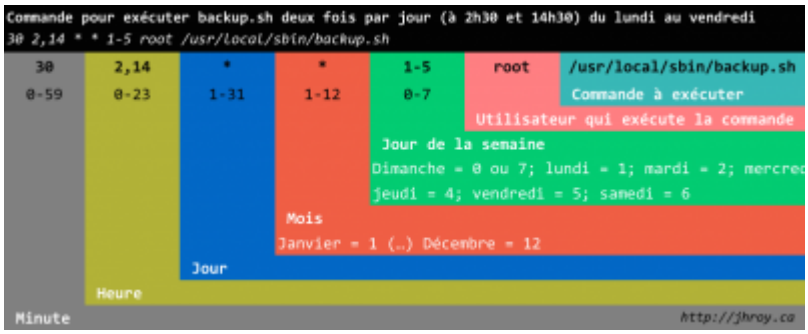
Par dessus tout, le script est la mémoire d'un procédé, d'une expertise, et permet de reproduire un résultat dans le futur.

Automatiser des tâches

Cron

Vixie's cron est un logiciel développé dans les années 1970 par Paul Vixie. L'idée était d'avoir un logiciel qui se réveille à chaque minute, et regarde dans une liste de fichiers s'il y a des tâches à exécuter à cet instant.

> cron est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance, ou selon un cycle défini à l'avance.




```
mm hh jj MMM JJJ tâche  > fichier-1  2> fichier-2
```

Références:

- (Wikipédia 2018)
- crontab guru

Développer des logiciels

GNU Make

Make est un logiciel qui sert à construire des fichiers ou des bibliothèques à partir de divers éléments, dont du code source. La principale différence entre Make et un script est qu'il va seulement exécuter des commandes si elles sont nécessaires. Make introduit donc la notion d'interdépendance entre les différentes commandes d'un programme.

Exemple d'un makefile:

```
# Mon premier Makefile

all: foobar.o main.c
        gcc -o main foobar.o main.c

foobar.o: foobar.c foobar.h
```

```
        gcc -c foobar.c -o foobar.o
```

La commande make apparaît dans la séquence traditionnelle de compilation et d'installation d'un logiciel depuis son code source.

```
#!/bin/bash
./configure
make
sudo make install
```

Référence:

- Compilation séparée et Make
- Utilisation de configure, make, make install

git

git est un système créé par Linus Torvalds qui permet de gérer les différentes versions de fichiers de manière organisée et collaborative. Le logiciel permet aussi d'utiliser un dépot centralisé où les membres d'une équipe peuvent publier leurs modifications à un projet. Le populaire site GitHub, récemment acquis par Microsoft, a aidé à populariser l'usage de git. Une alternative libre à GitHub est GitLab. Il est offert en tant que gratuiciel sur leur site web, et peut aussi être installé sur son propre serveur

Un flux de travail simple avec git:

```
#!/bin/bash
git clone https://gitlab.com/linuq/intro-ligne-commande-bash.git
git add nouveau_fichier
git commit -m "j'ai ajouté un nouveau fichier"
git push origin master
```

Références:

- git
- GitHub
- GitLab

Comprendre le fonctionnement du système d'exploitation

Périphériques

Bash donne accès directement à une abstraction des différents périphériques de l'ordinateur en tant que fichiers. On les retrouve dans le répertoire `/dev`.

Disques et mémoire

L'ensemble des disques disponibles est listée dans le répertoire `/dev/disk`

```
#!/bin/bash
ls -la /dev/disk
```

L'utilitaire `fdisk` permet de partitionner un disque dur. Ensuite, l'utilitaire `mkfs` permet de créer un système de fichiers sur chacune des partitions. La commande `free` vous permet de voir la quantité de mémoire vive disponible, et la commande `df` donne accès à l'utilisation de chacune des partitions.

Processus

Il est possible d'interagir directement avec l'exécution des processus via Bash.

- `ps` permet de listes tous les processus actifs selon un filtre donné en argument.
- `top` permet d'avoir une vue d'ensemble sur les processus qui sont exécutés par le système d'exploitation et d'agir sur ceux-ci.
- `kill` permet de tuer un processus, aussi simple que ça.

La combinaison suivante des commandes `ps` et `grep` permet de lister tous les processus exécutés qui contiennent `firefox` dans leur nom.

```
#!/bin/bash
ps -aux | grep firefox
```

Matériel d'apprentissage sélectionné

En français

- Guide avancé d'écriture des scripts Bash: Traduction française du manuel le plus complet pour Bash
- MOOC Maîtriser le shell Bash: Mooc développé par le gouvernement français, disponible sous licence CC-BY-NC-SA
- Administration système GNU/Linux: Diapositives d'une formation complète sur l'administration système disponible sous licence CC-BY-SA

En anglais:

- HakTip with Shannon Morse: Chaîne Youtube avec d'excellent vidéos très accessibles
- Wizard Zines by Julia Evans: Magazines et bandes dessinées sur des sujets informatiques et réseautique
- Ryan's tutorials: Plusieurs tutoriels interactifs sur la programmation

Références

Raymond, Eric Steven. 2003. “The Art of Unix Programming.” In. Addison-Wesley.

Wikipédia. 2018. “Cron — Wikipédia, L’encyclopédie Libre.” http://fr.wikipedia.org/w/index.php?title=Cron&oldid=152862426.