Introduction à la ligne de commande Linux/Unix

Atelier d'Isabelle Giguère pour BiblioTECH Jumpstart Program 2023

Cet atelier est largement inspiré de : https://librarycarpentry.org/lc-shell/aio.html

Résumé et configuration

Cet atelier a pour objectif de vous présenter le shell Unix. À la fin de la leçon, vous serez capable de : décrire les bases de l'interpréteur de commandes Unix ; expliquer pourquoi et comment utiliser la ligne de commande utiliser les commandes de l'interpréteur de commandes pour travailler avec les répertoires et les fichiers ; utiliser les commandes de l'interpréteur de commandes pour trouver et manipuler des données.

Pré requis

Pour compléter cette leçon, vous aurez besoin d'un environnement shell de type Unix - voir Configuration. Vous devrez également télécharger le fichier shell_lesson.zip de GitHub sur votre bureau et l'extraire (une fois le fichier décompressé/extrait, vous devriez obtenir un dossier appelé "shell_lesson").
Pour participer à cet atelier, vous aurez besoin d'un environnement shell de type Unix. Plus précisément, nous utiliserons Bash (Bourne Again Shell) qui est standard sur Linux et macOS. Les utilisateurs de macOS Catalina auront zsh (Z shell) comme version par défaut. Même si vous êtes un utilisateur de Windows, l'apprentissage de Bash vous ouvrira un ensemble d'outils puissants sur votre machine personnelle, en plus de vous familiariser avec 'interface distante standard utilisée sur presque tous les servers et super ordinateurs.

Configuration

Bash est l'interpréteur de commandes par défaut de la plupart des distributions Linux et de macOS. Les utilisateurs de Windows devront installer Git Bash pour bénéficier d'un environnement de type Unix.

Dans Linux : L'interpréteur de commandes par défaut est généralement Bash, mais si votre machine est configure différemment, vous pouvez l'exécuter en ouvrant un terminal > et en tapant bash. Il n'est pas nécessaire d'installer quoi que ce soit. Recherchez Terminal dans vos applications pour lancer l'interpréteur de commandes Bash.

macOS : Bash est l'interpréteur de commandes par défaut dans toutes les versions de macOS antérieures à Catalina, vous n'avez pas besoin d'installer quoi que ce soit. Ouvrez Terminal et à partir de > /Applications/Utilitaires ou de la recherche Spotlight pour lancer l'interpréteur de commandes Bash. zsh est l'interpréteur de commandes par défaut dans Catalina.

Windows : Sous Windows, CMD ou PowerShell sont normalement les environnements shell par défaut. Ceux-ci utilisent une syntaxe et un ensemble 'applications propres aux systèmes Windows et sont incompatibles avec les utilitaires Unix plus largement utilisés. Cependant, un shell Bash peut être installé sur Windows pour fournir un environnement de type Unix. Pour cette leçon, nous proposons d'utiliser Git Bash, qui fait partie du paquet >Git for Windows:

Télécharger la dernière version de l'installateur de Git pour Windows pour Windows.
Double-cliquez sur le fichier .exe pour exécuter le programme 'installation (par exemple, Git-2.41.0.3-64-bit.ex) en utilisant les paramètres par défaut.
Une fois installé, ouvrez le shell en sélectionnant Git Bash dans le menu Démarrer (dans le dossier Git).

Il existe également des solutions plus avancées pour exécuter des commandes Bash sous Windows. Un outil de ligne de commande de l'interpréteur de commandes Bash est disponible pour Windows 10, que vous pouvez utiliser si vous activez l'option Sous-système Windows pour Linux. En outre, vous pouvez exécuter des commandes Bash sur un ordinateur ou un server distant qui dispose déjà d'un shell Unix, à partir de vote machine Windows. Cette opération s'effectue généralement à l'aide d'un client Secure Shell (SSH). L'un de ces clients, disponible gratuitement pour les ordinateurs Windows, est PuTTY.

Si vous rencontrez des problèmes, The Carpentries maintiennent une page wiki "Configuration Problems and Solutions" qui peut vous aider.

Fichiers de données

Vous devez télécharger certains fichiers pour suivre cette leçon:
Téléchargez shell_lesson.zip et déplacez le fichier sur vote bureau.
Décompresser/extraire le fichier (demandez à vote formateur si vous avez besoin d'aide pour cette étape). Vous devriez obtenir un nouveau dossier appelé shell_lesson sur vote bureau.
Ouvrez le terminal et tapez ls suivi de la touche entrée.

```
$ ls
```

Une liste de fichiers et de dossiers devrait s'afficher dans vote répertoire actuel.
Tapez ensuite:

```
$ pwd
```

Cette commande vous montrera où vous vous trouvez dans votre système de fichiers, qui devrait maintenant être votre répertoire personnel. Dans cet atelier, vous en apprendrez plus sur les commandes ls, pwd et sur la manière de travailler avec les données du dossier shell_lesson.

1. Qu’est-ce que le 'shell'?

Qu'est-ce que le shell et pourquoi l'utiliser?

Si vous avez déjà eu à traiter de grandes quantités de données ou de fichiers numériques dispersés sur vote ordinateur ou sur un server distant, vous savez que la copie, le déplacement, le renommage, le comptage, la recherche ou tout autre traitement manuel de ces fichiers put prendre énormément de temps et être source d'erreurs. Heureusement, il existe un outil extraordinairement puissant et flexible conçu à cette fin.

Le shell (parfois appelé "interpréteur de commandes Unix", du nom du système 'exploitation ou il a été développé pour la première fois) est un programme qui vous permet d'interagir avec vote ordinateur à l'aide de commandes textuelles saisies. Il s'agit de 'interface principale utilisée sur Linux et les systèmes basés sur Unix, tels que macOS, et peut être installé en option sur d'autres systèmes d'exploitation tels que Windows.

C'est l'exemple parfait d'une "interface de ligne de commande", où les instructions sont données à l'ordinateur en tapant des commandes, et où l'ordinateur répond en exécutant une tâche ou en générant un résultat. Ce résultat est généralement affichée à l'écran, mais il peut être dirigé vers un fichier ou même vers d'autres commandes, ce qui permet de créer de puissantes chaines d'actions ave très peu d'efforts.

L'utilisation d'un shell resemble parfois plus à de la programmation qu'à l'utilisation d'une souris. Les commandes sont courtes (souvent quelques caractères seulement), leurs noms sont souvent énigmatiques et leurs résultats sont des lignes de texte plutôt que quelque chose de visuel comme un graphique. D'un autre côté, avec seulement quelques frappes, l'interpréteur de commandes vous permet de combiner des outils existants en de puissants pipelines et de traiter automatiquement de grands volumes de données. Cette automatisation vous rend non seulement plus productif, mais elle améliore également la reproductibilité de vos flux de travail en vous permettant de les enregistrer et de les répéter à l'aide de quelques commandes simples. Comprendre les bases du shell constitue une base utile pour apprendre à programmer, puisque certains des concepts que vous apprendrez ici - tels que les boucles, les valeurs et les variables - se traduiront en programmation.

Le shell est l'un des environnements de programmation les plus productifs jamais créés. Une fois maîtrisé, vous pouvez l'utiliser pour expérimenter différentes commandes de manière interactive, puis utiliser ce que vous avez appris pour automatiser vote travail.

Dans cet atelier, nous introduirons l'automatisation des tâches en examinant comment les données peuvent être manipulées, comptées et exploitées à l'aide du shell. Nous couvrirons un petit nombre de commandes de base, qui constitueront

les éléments de base à partir desquels des commandes plus complexes pourront être construites pour s'adapter à vos données ou à votre projet. Même si vous ne programmez pas vous-même ou si votre travail ne fait pas appel à la ligne de commande, il peut être utile de connaître les bases de l'interpréteur de commandes.

Où est mon shell?

L'interpréteur de commandes est un programme qui est généralement lancé sur votre ordinateur de la même manière que n'importe quel autre programme. Cependant, il existe de nombreux types d'interpréteurs de commandes portant des noms différents, qui peuvent être ou non déjà installés. Le shell est au cœur des ordinateurs basés sur Linux, et les machines macOS sont livrées avec Terminal, un programme d'interpréteur de commandes. Pour les utilisateurs de Windows, des shells populaires tels que Cygwin ou Git Bash fournissent une interface de type Unix, mais peuvent nécessiter une installation séparée. Dans Windows 10, le sous-système Windows pour Linux donne également accès à un outil de ligne de commande de 'interpréteur de commandes Bash.

2. Navigation dans le système de fichiers

Nous commencerons par les bases de la navigation à l'aide du shell.

Commençons par ouvrir le shell. Cela se traduit probablement par l'affichage d'une fenêtre noire ou blanche avec un curseur clignotant à côté d'un signe de dollar. Il s'agit de notre ligne de commande, et le $ est 'invite de commande qui indique que le système est prêt à recevoir nos données. L'apparence de l'invite varie d'un système à l'autre, en fonction de la façon dont le système a été configuré. D'autres invites courantes comprennent les signes % ou #, mais nous utiliserons $ dans cette leçon pour représenter l'invite en général.

Lorsque vous travaillez dans le shell, vous vous trouvez toujours quelque part dans le système de fichiers (filesystem) de l'ordinateur, dans un quelconque dossier (répertoire). Nous allons donc commencer par savoir où nous nous trouvons en utilisant la commande pwd, que vous pouvez utiliser chaque fois que vous n'êtes pas sûr de votre position. Elle signifie "imprimer le répertoire de travail" et le résultat de la commande est imprimé sur la sortie standard, c'est-à-dire l'écran.

Saisissons pwd et appuyons sur la touche Entrée pour exécuter la commande (Notez que le signe $ est utilisé pour indiquer une commande à saisir à l'invite de commande, mais que nous ne saisissons jamais le signe $ lui-même, seulement ce qui suit).

Le résultat sera un chemin 'accès à votre répertoire personnel. Vérifions si nous le reconnaissons en examinant le contenu du répertoire. Pour ce faire, nous utilisons la commande ls. Cette commande signifie "liste" et le résultat est une impression

```
$ pwd
/Users/armide
$ ls
Desktop Downloads       Movies  Pictures Public
Documents       Library Music           Postman         marcedit35      shell_lesson
```

de tout le contenu du répertoire (ici sur un Mac) :

Il se peut que nous voulions plus d'informations qu'une simple liste de fichiers et de répertoires. Nous pouvons obtenir ces informations en spécifiant différents options ('flags'), également appelés paramètres ou, le plus souvent, arguments) pour accompagner nos commandes de base. Les arguments modifient le fonctionnement de la commande en indiquant à l'ordinateur le type de sortie ou de manipulation que nous souhaitons.

Si nous tapons ls -l et appuyons sur Retour, l’ordinateur renvoie une liste de fichiers contenant des informations similaires à celles que nous trouverions dans notre Finder (Mac) ou notre Explorer (Windows) : la taille des fichiers en octets, la date de création ou de dernière modification et le nom du fichier. Dans 'usage courant, nous sommes plus habitués à des unités de mesure comme les kilo-octets, les méga-octets et les giga-octets. Heureusement, il existe une autre option -h qui, lorsqu'il est utilisé avec l'option -l, utilise les suffixes d’unité: Octet, Kilo-octet, Méga-octet, Giga-octet, Téra-octet et Péta-octet afin de réduire le nombre de chiffres à trois ou moins.

```
$ ls -l
total 0
drwx------@  43 armide  staff  1376 24 jul 15:04 Desktop
drwx------@ 124 armide  staff  3968 24 jul 14:10 Documents
drwx------@  17 armide  staff   544 24 jul 11:23 Downloads
drwx------@  92 armide  staff  2944  8 jul 19:31 Library
drwx------    6 armide  staff   192  3 avr 18:21 Movies
drwx------+   7 armide  staff   224 16 jui 10:46 Music
drwx------+   9 armide  staff   288 16 jui 12:03 Pictures
drwxr-xr-x    3 armide  staff    96  8 jui 16:57 Postman
drwxr-xr-x+   5 armide  staff   160 24 jul 14:10 Public
drwxr-xr-x   14 armide  staff   448 16 jui 10:46 marcedit35
drwxr-xr-x   12 armide  staff   384 24 jul 14:47 shell_lesson
```

Cependant, ls -h ne fonctionnera plus seul. Lorsque nous voulons combiner deux options, nous pouvons simplement les exécuter ensemble. Ainsi, en tapant Is -lh et en appuyant sur entrée, nous obtenons une sortie dans un format lisible par l'humain (note : l’ordre ici n'a 'pas d’importance).

```
$ ls -lh
total 0
drwx------@  43 armide  staff   1,3K 24 jul 15:04 Desktop
drwx------@ 124 armide  staff   3,9K 24 jul 14:10 Documents
drwx------@  17 armide  staff   544B 24 jul 11:23 Downloads
drwx------@  92 armide  staff   2,9K  8 jul 19:31 Library
drwx------    6 armide  staff   192B  3 avr 18:21 Movies
drwx------+   7 armide  staff   224B 16 jui 10:46 Music
drwx------+   9 armide  staff   288B 16 jui 12:03 Pictures
drwxr-xr-x    3 armide  staff    96B  8 jui 16:57 Postman
drwxr-xr-x+   5 armide  staff   160B 24 jul 14:10 Public
drwxr-xr-x   14 armide  staff   448B 16 jui 10:46 marcedit35
drwxr-xr-x   12 armide  staff   384B 24 iul 14:47 shell lesson
```

Nous avons passé beaucoup de temps dans notre répertoire d'origine. Allons ailleurs. Nous pouvons le faire à l'aide de la commande cd ou Change Directory (Remarque : sous Linux, on doit prendre en compte la casse des noms de fichier ou répertoire.

Remarquez que la commande n'a rien produit. Cela signifie qu'elle a été exécutée avec succès. Vérifions en utilisant pwd.

Si quelque chose n'avait pas fonctionné, la commande vous l'aurait dit. Testons cela en essayant de nous déplacer dans un répertoire inexistant:

```
$ cd "my first directory"
bash: cd: my first directory: No such file or directory
```

`$ cd "my first directory"`
Remarquez que nous avons entouré le nom de guillemets. Les arguments donnés à toute commande du shell sont séparés par des espaces, de sorte que l'utilisation de guillemets (simples ou doubles) permet d'indiquer que nous voulons dire "une seule chose appelée "my first directory", et non pas six chose différentes.

Nous avons maintenant vu comment nous pouvons descendre dans notre structure de répertoires (c'est-à-dire dans des répertoires plus imbriqués). Si nous voulons revenir en arrière, nous pouvons taper `cd ..` Cela nous fait remonter d'un répertoire et nous ramène à notre point de départ. Si jamais nous sommes complètement perdus, la commande `cd` sans arguments nous ramènera directement à la racine de notre répertoire, là où nous avons commencé.

Répertoire précédent

Pour aller et venir entre deux répertoires, utilisez `cd -`.

Explorer et déplacez vous

Déplacez-vous sur l'ordinateur, habituez-vous à entrer et sortir des répertoires, voyez comment les différents types de fichiers apparaissent dans le shell Unix. Veillez à utiliser les commandes pwd et cd, ainsi que les différentes options de la commande ls apprises jusqu'à présent.

Il est très important de pouvoir naviguer dans le système de fichiers pour utiliser efficacement les commandes du shell. En devenant plus à l'aise, nous pouvons atteindre très rapidement le répertoire que nous voulons.

Obtenir de l'aide

La commande man permet d'appeler la page de documentation d'une commande de l'interpréteur de commandes. Par exemple, `man ls` affiche tous les arguments disponibles - ce qui vous évite de les mémoriser tous!
Essayez ceci pour chaque commande que vous avez apprise jusqu'à présent.
Utilisez la barre d'espacement pour naviguer dans les pages du manuel. Utilisez q à tout moment pour quitter.

Remarque : cette commande est réservée aux utilisateurs de Mac et de Linux. Elle ne fonctionne pas directement pour les utilisateurs de Windows. Si vous utilisez Windows, vous pouvez rechercher la commande shell sur http://man.he.net/ et consulter la page de manuel associée. Sur certains systèmes, le nom de la commande suivi de --help fonctionne, par exemple ls --help.

En outre, le manuel répertorie les commandes individuellement. Par exemple, bien que l'option -h ne puisse être utilisée qu'avec l'option -l, elle est répertoriée comme -h dans le manuel, et non comme -lh.

Découvrez les commandes ls avancées

Découvrez, à l'aide de la page de manuel, comment lister les fichiers d'un répertoire en fonction de leur taille. Essayez-le dans différents répertoires. Pouvez-vous le combiner avec l'argument -l que vous avez appris précédemment?

Ensuite, découvrez comment vous pouvez classer une liste de fichiers en fonction de leur date de dernière modification. Essayez de classer les fichiers dans différents répertoires.

Savoir où l'on se trouve dans la structure des répertoires est essentiel pour travailler avec le shell.

3. Working with files and directories

Outre la navigation dans les répertoires, nous pouvons interagir avec les fichiers sur la ligne de commande : nous pouvons les lire, les ouvrir, les exécuter et même les modifier. En fait, il n'y a pas de limite à ce que nous pouvons faire avec le shell, mais même les utilisateurs expérimentés de commandes optent encore pour des interfaces graphiques pour de nombreuses tâches, telles que l'édition de documents textuels formatés (Word ou OpenOffice), la navigation sur le web, l'édition d'images, etc. Mais si nous voulions effectuer le même recadrage sur des centaines d'images, par exemple les pages d'un live numérisé, nous pourrions automatiser ce travail de recadrage en utilisant des commandes du shell.

Avant de commencer, nous allons utiliser cd pour vérifier où nous en sommes. L'utilisation périodique de cd pour visualiser vos options est utile pour s'orienter.

Nous allons essayer quelques méthodes de base pour interagir avec les fichiers. Commençons par le répertoire shell_lesson sur votre bureau.

```
$ cd
$ cd shell_lesson
$ pwd
/Users/armide/shell_lesson
$ mkdir firstdir
```

Nous allons créer un nouveau répertoire et nous y installer :
Ici, nous avons utilisé la commande mkdir (qui signifie "créer des répertoires") pour créer un répertoire nommé "firstdir". Nous nous sommes ensuite déplacés dans ce répertoire à l'aide de la commande cd.

Mais attendez! Il existe une astuce pour accélérer les choses. Remontons d'un répertoire.

`$ cd ..`
Au lieu de taper `cd firstdir`, essayons de taper `cd f` et d'appuyer sur la touche

```
$ cd firstdir
$ pwd
/Users/armide/shell_lesson/firstdir
```

Tab. Nous remarquons que le shell complète la ligne `cd firstdir/`.

Touche TAB pour l'auto-complétion

En appuyant sur la touche TAB à tout moment à la ligne de commandes, ce dernier tentera de compléter automatiquement la ligne en fonction des fichiers ou des sous-répertoires du répertoire actuel. Lorsque deux fichiers ou plus contiennent les mêmes caractères, l'auto-complétion ne se remplira que jusqu'au premier point de différence, après quoi nous pouvons ajouter d'autres caractères et réessayer d'utiliser la touche tabulation. Nous vous encourageons à utiliser cette méthode tout au long de la journée pour voir comment elle se comporte (car elle permet d'économiser beaucoup de temps et d'efforts !).

Si vous êtes dans firstdir, utilisez cd .. pour revenir au répertoire shell_ lesson.

On y trouve des copies de deux lives du domaine public téléchargés à partir du Projet Gutenberg, ainsi que d'autres fichiers dont nous parlerons plus tard.

Les fichiers 829-0.txt et 33504-0.txt contiennent le contenu des lives #829 et #33504 sur le Projet Gutenberg. Mais nous avons oublié quels livres, alors nous

```
$ cat 829-0.txt
```

Utilisons la commande cat pour lire le texte du premier fichier.
La fenêtre du terminal défile le livre en cascade (il s'affiche sur votre écran), nous laissant avec une nouvelle invite et les dernières lignes du fichier au-dessus de cette invite.

Souvent, nous voulons juste jeter un coup d'œil rapide à la première ou à la dernière partie d'un fichier pour nous faire une idée de son contenu. Pour ce faire,

```
$ head 829-0.txt
The Project Gutenberg eBook, Gulliver's Travels, by Jonathan Swift


This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org
```

le shell met à notre disposition les commandes head et tail.

Il s'agit d'une vue des dix premières lignes, tandis que tail 829-0.txt offre une perspective sur les dix dernières lignes:

```
$ tail 829-0.txt

Most people start at our Web site which has the main PG search facility:

    http://www.gutenberg.org

This Web site includes information about Project Gutenberg-tm,
including how to make donations to the Project Gutenberg Literary
Archive Foundation, how to help produce our new eBooks, and how to
```

Si dix lignes ne suffisent pas (ou que c'est trop), consultons `man head` (ou `head -help` sous Windows) pour voir s'il existe une option permettant de spécifier le nombre de lignes à obtenir (c'est le cas : `head -n 20` imprimera 20 lignes).

Une autre façon de naviguer dans les fichiers est d'en afficher le contenu un écran à la fois. Tapez `less 829-0.txt` pour voir le premier écran, la barre d'espacement pour voir l'écran suivant et ainsi de suite, puis a pour quitter (revenir à 'invite de commande).

Comme beaucoup d'autres, les commandes cat, head, tail et less peuvent prendre plusieurs arguments et peuvent fonctionner avec n'importe quel nombre de fichiers. Nous allons voir comment nous pouvons obtenir les premières lines de plusieurs fichiers à la fois. Afin d'économiser des frappes, introduisons d'abord une astuce très utile.

Réutilisation des commandes

À l’invite de commande, appuyez sur la touche fléchée vers le haut et remarquez que la commande précédente que vous avez tapée apparaît devant votre curseur. Vous pouvez continuer à appuyer sur la flèche vers le haut pour parcourir les commandes précédentes. La flèche vers le bas permet de revenir à la commande la plus récente. Il s'agit là d'une autre fonction importante qui permet d'économiser du temps et que nous utiliserons souvent.

Appuyez sur la flèche vers le haut jusqu'à ce que vous arriviez à la commande `head`

```
$ head 829-0.txt 33504-0.txt
==> 829-0.txt <==
The Project Gutenberg eBook, Gulliver's Travels, by Jonathan Swift


This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org



==> 33504-0.txt <==
The Project Gutenberg EBook of Opticks, by Isaac Newton

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org
```

`829-0.txt`. Ajoutez un espace, puis 33504-0.txt (vous vous souvenez de votre ami TAB : tapez 3 suivi de TAB pour obtenir 33504-0.txt) et pour produire la commande suivante.

Cependant, si nous avions beaucoup de fichiers, il serait fastidieux d'entrer tous les noms de fichiers. Heureusement, la ligne de commande prend en charge les caractères de substitution. Les ? (correspond à un seul caractère) et * (correspond à zéro ou plusieurs caractères). Nous pouvons utiliser le caractère * pour écrire la commande head ci-dessus de manière plus compacte : `head *.txt`.

Troncature

Les caractères de substitution (ou troncature) sont une caractéristique du shell et fonctionnent donc avec 'importe quelle commande. Le shell développe ces caractères en une liste de fichiers ou de répertoires avant l'exécution de la commande, et la commande ne voit jamais les caractères génériques. À titre d'exception, si une expression ne correspond à aucun fichier, Bash transmettra l'expression telle quelle en tant que paramètre de la commande. Par exemple, si

l'on tape `ls *.pdf`, un message d'erreur s'affiche, indiquant qu'il 'existe pas de fichier appelé *.pdf.

Déplacer, copier et supprimer des fichiers

Il se peut également que nous voulions changer le nom du fichier pour quelque chose de plus descriptif. Nous pouvons le déplacer vers un nouveau nom en utilisant la commande mv (move), en lui donnant l'ancien nom comme premier argument et le nouveau nom comme second argument :

```
mv 829-0.txt gulliver.txt
```

C'est 'équivalent de la fonction "renommer un fichier."

Par la suite, lorsque nous exécuterons la commande ls, nous verrons qu'il s'appelle

```
$ ls
2014-01-31_JA-africa.tsv 2014-01-31_JA-america.tsv 2014-01_JA.tsv 2014-02-02_JA-britain.tsv 201403160_01_text.json 33504-0.txt diary.html firstdir gulliver.txt loans_2022_2023
```

désormais gulliver.txt :

- Au lieu de déplacer un fichier, vous pouvez vouloir copier un fichier (faire un double), par exemple pour faire une sauvegarde avant de modifier un fichier. Tout comme la commande mv, la commande cp prend deux arguments : l'ancien nom et le nouveau nom. Comment feriez-vous une copie du fichier gulliver.txt appelée gulliver-backup.txt ? Essayez-le!

- Renommer un répertoire fonctionne de la même manière que renommer un fichier. Essayez d'utiliser mv pour renommer le répertoire 'firstdir' à 'backup'.

- Si le dernier argument que vous donnez à la commande mv est un répertoire et non un fichier, le fichier donné en premier argument sera déplacé dans ce répertoire. Essayez d'utiliser mv pour déplacer le fichier gulliver-backup.txt dans le dossier backup.

- La commande history permet d'afficher une liste de toutes les commandes que vous avez saisies au cours de la session actuelle.

Vous pouvez également effectuer une recherche dans l'historique. Appuyez sur Ctrl + r, puis commence à taper n'importe quelle partie de la commande que vous recherchez. La commande précédente s'affiche automatiquement. Appuyez sur Entrée pour exécuter la commande à nouveau, ou appuyez sur les touches fléchées pour commencer à modifier la commande. Si plusieurs commandes passées contiennent le texte que vous avez saisi, vous pouvez appuyer plusieurs fois sur Ctrl + r pour les parcourir. Si vous ne trouvez pas ce que vous cherchez, utilisez Ctrl + c pour revenir à l'invite. Si vous voulez sauvegarder vote historique, peut-être pour extraire quelques commandes à partir desquelles vous pourrez construire un script plus tard, vous pouvez le faire avec history > history.txt. Cette commande enregistre tout l'historique dans un fichier texte appelé history.txt, que vous pourrez éditer ultérieurement.
Pour rappeler une commande de l'historique, notez le numéro de la commande, par exemple 2045. Rappelez la commande en entrant `!2045`. Cela exécutera la commande.

Utilisation de la commande echo

La commande echo imprime simplement un texte que vous spécifiez. Essayez-le echo 'Library Carpentry is awesome!'. Intéressant, n'est-ce pas?

Vous pouvez également spécifier une variable. Tapez d'abord `NAME=` suivi de votre nom et appuyez sur entrée. Then type `echo "$NAME est étudiant fantastique"` et appuyez sur la touche Entrée. Que se passe-t-il?

Vous pouvez combiner du texte et des commandes shell en utilisant echo, par exemple, la commande pwd que vous avez apprise plus tôt. Pour ce faire, vous devez entourer une commande shell de $(et), par exemple $(pwd).
Essayez maintenant ce qui suit :
`echo "Finalement, il fait soleil, le " $(date)`

Notez que la sortie de la commande date est imprimée avec le texte que vous avez spécifié. Vous pouvez essayer la même chose avec d'autres commandes que vous avez apprises jusqu'à présent.

Pourquoi pensez-vous que la commande echo est très important dans l'environnement shell ?

Enfin, parlons de la suppression. Nous ne l'utiliserons pas pour l'instant, mais si vous souhaitez supprimer un fichier, pour quelque raison que ce soit, la commande est rm (remove).

En utilisant la troncature, nous pouvons même supprimer un grand nombre de fichiers. En ajoutant l'option -r, nous pouvons supprimer des dossiers avec tout leur contenu.

Contrairement à la suppression à partir de notre interface graphique, il n'y a pas d'avertissement, pas de corbeille de recyclage dans laquelle vous pourrez récupérer les fichiers et pas de possibilité d'annulation! C'est pourquoi il convient d'être très prudent avec rm et extrêmement prudent avec rm -r.

Points clés

La ligne de commande put être utilisé pour copier, déplacer et combiner plusieurs fichiers.

4. Automatiser des commandes avec des boucles (loops)

Écrire une boucle

Les boucles sont essentielles à l'amélioration de la productivité, car elles nous permettent d'exécuter des commandes de manière répétitive. A l'instar des caractères de substitutions et de la complétion par TAB, l'utilisation de boucles permet également de réduire le nombre de commandes saisies (et de fautes de frappe). Supposons que nous ayons plusieurs centaines de fichiers de documents nommés projet_1825.txt, projet_1863.txt, XML_projet.txt, etc. Nous souhaitons modifier ces fichiers, mais aussi enregistrer une version des fichiers originaux, en nommant les copies sauvegarde_projet_1825.txt, etc.

Pour ce faire, nous pouvons utiliser une boucle. Voici un exemple simple qui crée une copie de sauvegarde de quatre fichiers texte à tour de rôle.

Créons d'abord ces fichiers :

```
$ touch a.txt b.txt c.txt d.txt
```

Cela créera quatre fichiers vides portant ces noms.

Nous allons maintenant utiliser une boucle pour créer une version de sauvegarde de ces fichiers. Voyons d'abord la forme générale d'une boucle.

```
for element in liste_elements
do
   operation_utilisant $element
```

```
$ touch a.txt b.txt c.txt d.txt
$ for filename in ?.txt
> do
>   echo "filename"
>   cp "filename" backup_"$filename"
> done
a.txt
b.txt
c.txt
d.txt
```

```
done
```

Lorsque l'interpréteur de commandes voit le mot clé for, il sait qu'il doit répéter une commande (ou un groupe de commandes) pour chaque élément d'une

liste. À chaque itération, le nom de chaque élément est assigné séquentiellement à la variable de la boucle et les commandes à l'intérieur de la boucle sont exécutées avant de passer à l'élément suivant de la liste. A l'intérieur de la boucle, nous demandons la valeur de la variable en plaçant $ devant elle. Le $ indique à Au shell de traiter la variable comme un nom de variable et de substituer sa valeur à sa place, plutôt que de la traiter comme du texte ou une commande externe.

Substitutions de variables entre guillemets doubles

Comme les noms de fichiers réels contiennent souvent des spaces blancs, nous mettons $filename entre guillemets doubles ("). Si nous ne le faisions pas, le shell considérerait l'espace blanc dans un nom de fichier comme un séparateur entre deux noms de fichiers différents, ce qui entraînerait généralement des erreurs. Par conséquent, il est préférable et généralement plus sûr d'utiliser "$...", à moins que vous ne soyez absolument certain qu'aucun élément comportant un espace blanc ne puisse jamais entrer dans votre variable de boucle.

Dans l’exemple précédent, la liste est constituée de quatre noms de fichiers : "a.txt", "b.txt", "C. txt "et "d.txt. "Chaque fois que la boucle est itérée, elle attribue un nom de fichier à la variable filename et exécute la commande cp. Lors du premier passage dans la boucle, $filename est a.txt. Le shell affiche le nom du fichier à l'écran, puis exécute la commande cp sur a.txt, (parce que nous lui avons demandé d'afficher chaque nom de fichier au fur et à mesure qu'il parcourt la boucle). Pour la deuxième itération, $filename devient b.txt. Cette fois, l'interpréteur de commandes affiche le nom de fichier b.txt à l'écran, puis exécute cp sur b.txt. La boucle effectue les mêmes opérations pour c.txt, puis pour d.txt et, comme la liste ne contient que ces quatre éléments, l'interpréteur de commandes quitte la boucle for à ce stade.

Suivre l’invite du shell

La ligne de commande passe du $ à > et vice-versa au fur et à mesure que nous tapons dans notre boucle. La deuxième invite, >, est différente pour nous rappeler que nous n avons pas encore fini de taper une commande complète. Un point-virgule, ;, peut être utilisé pour séparer deux commandes écrites sur une seule ligne.

Même symboles, significations différentes

Ici, > est utilisé comme invite du shell, mais > peut également être utilisé pour rediriger la sortie d'une commande (c'est-à-dire l'envoyer ailleurs, par exemple dans un fichier, au lieu d'afficher la sortie dans le terminal) - nous utiliserons la redirection dans la partie 5. De même, $ est utilisé comme invite du shell mais, comme nous l'avons vu précédemment, il est également utilisé pour demander à l'interpréteur de commandes de définir une variable pour lui attribuer une valeur.

Si le shell affiche > ou $, c'est qu'il s'attend à ce que vous tapiez quelque chose, et le symbole est une invite.

Si vous tapez > dans l'interpréteur de commandes, il s'agit d'une instruction que vous lui donnez pour rediriger la sortie d'un résultat.

Si vous tapez $ dans le shell, il s'agit d'une instruction que vous lui donnez pour obtenir la valeur d'une variable.

Nous avons appelé la variable de cette boucle filename (nom de fichier) afin de rendre son objectif plus clair pour les lecteurs humans. L'interpréteur de commandes lui-même ne se préoccupe pas du nom de la variable.

Complétez les blancs dans la boucle for ci-dessous pour imprimer le nom, la première ligne et la dernière ligne de chaque fichier texte dans le répertoire actuel.

```
__ file in *.txt
_
        echo "_file"
        head -n 1 _____
        ___ _ _ _____
___
```

Nous exécuterons une autre boucle dans la section suivante Compter et extraire avec le shell.

Exécuter une boucle à partir d'un script BASH

Au lieu d'exécuter la boucle ci-dessus à la ligne de commande, vous pouvez l'enregistrer dans un fichier et l'exécuter à partir de la ligne de commande sans avoir à réécrire la boucle. C'est ce qu'on appelle un script BASH, qui est un fichier

texte contenant une série de commandes comme la boucle que vous avez créé ci-dessus.
Dans l'exemple de script ci-dessous, la première ligne du fichier contient ce que l'on appelle un Shebang (# !) suivi du chemin d'accès au shell (ou programme) qui exécutera le reste des lignes du fichier (/bin/bash). La deuxième ligne montre comment les commentaires sont faits dans les scripts. Cela vous donne plus 'informations sur ce que fait le script. Les lignes restates contiennent la boucle que vous avez créée ci-dessus. Vous pouvez créer ce fichier dans le même répertoire que vous avez utilisé pour la leçon et en utilisant l'éditeur de texte de votre choix (par exemple vi), mais lorsque vous enregistrez le fichier, assurez-vous qu'il porte l'extension .bash (par exemple mon_premier_script.bash). Une fois cela fait, vous pouvez exécuter le script Bash en tapant la commande bash et le nom du fichier dans la ligne de commande (par exemple bash mon_premier script bash.sh).

```
#!/bin/bash
# This script loops through .txt files, returns the file name, first line, and last line of the file
for file in *.txt
do
        echo $file
        head -n 1 $file
        tail -n 1 $file
done
```

Copiez le fichier my_first_bash_script.sh, vous pourrez le retravailler plus tard.

Pour en savoir plus sur les scripts Bash, voir Bash Scripting Tutorial - Ryans Tutorials.

Points clés

La boucle est la base pour travailler plus intelligemment avec la ligne de commande.

Les boucles nous aident à faire la même chose (ou des choses similaires) à un ensemble d'éléments.

Avant de poursuivre, nous allons supprimer les fichiers vides que nous venons de

```
$ ls ?.txt
a.txt     b.txt     c.txt     d.txt
$ rm ?.txt
$ ls backup_?.txt
backup_a.txt     backup_b.txt     backup_c.txt     backup_d.txt
```

créer.

5. Compter et extraire avec le shell

Maintenant que vous savez comment naviguer dans le shell, nous allons apprendre à compter et à extraire des données à 'aide de quelques commandes standard. Bien qu'il soit peu probable que ces commandes révolutionnent vote travail, elles sont très polyvalentes et vous aideront à travailler dans le shell et à apprendre à coder. Les commandes reproduisent également les types d'utilisation que les utilisateurs de la bibliothèque peuvent faire des données de la bibliothèque.

Compter et trier

Nous commencerons par compter le contenu des fichiers à la ligne de commandes Unix. Le shell Unix permet de générer rapidement des comptages à partir de plusieurs fichiers, ce qui est difficile à réaliser à l'aide des interfaces graphiques des suites bureautiques standard.

Commençons par nous rendre dans le répertoire qui contient nos données à l'aide de la commande cd :

```
$ cd shell_lesson
```

N'oubliez pas que si vous ne savez pas où vous vous trouvez dans la structure de vos répertoires, utilisez la commande pwd pour le savoir:

```
$ pwd
/Users/armide/Desktop/shell_lesson
```

```
$ ls -lhS
total 618696
-rw-r--r--@ 1 armide  staff   125M 10 jui  2015 2014-01_JA.tsv
-rw-r--r--  1 armide  staff    26M 24 jul 14:27 loans_2022_2023
-rw-r--r--@ 1 armide  staff   7,4M  7 oct  2021 2014-01-31_JA-america.tsv
-rw-r--r--@ 1 armide  staff   3,6M  7 oct  2021 2014-01-31_JA-africa.tsv
-rw-r--r--@ 1 armide  staff   1,4M  7 oct  2021 2014-02-02_JA-britain.tsv
-rw-r--r--@ 1 armide  staff   588K 19 jul 12:50 gulliver.txt
-rw-r--r--@ 1 armide  staff   582K  7 oct  2021 33504-0.txt
-rw-r--r--@ 1 armide  staff   383K  7 oct  2021 201403160_01_text.json
```

Vérifions quels fichiers se trouvent dans le répertoire et quelle est leur taille : Dans cette section, nous nous concentrerons sur l'ensemble de données 2014-01_JA.tsv, qui contient des métadonnées 'articles de journaux, et sur les trois fichiers .tsv dérivés de l'ensemble de données original. Chacun de ces trois fichiers .tsv comprend toutes les données où un mot-clé tel que africa ou america apparaît dans le champ 'Title 'de 2014-01_JA.tsv.

```
$ wc *.tsv
   13712  511261 3773660 2014-01-31_JA-africa.tsv
   27392 1049601 7731914 2014-01-31_JA-america.tsv
  507732 17606310 131122144 2014-01_JA.tsv
    5375  196999 1453418 2014-02-02_JA-britain.tsv
  554211 19364171 144081136 total
```

Fichiers CSV et TSV

CSV (Comma-separated values) est un format de texte brut courant pour le stockage de données tabulaires, où chaque enregistrement occupe une ligne et où les valeurs sont séparées par des virgules. Le format TSV (Tab-separated values) est identique, mais les valeurs sont séparées par des tabulations plutôt que par des virgules. Le terme CSV est parfois utilisé pour désigner à la fois CV, TSV et leurs variantes, ce qui put prêter à confusion. La simplicité des formats les rend parfaits pour l'échange et l'archivage. Ils ne sont pas liés à un programme spécifique (contrairement aux fichiers Excel, par exemple, il n'y a pas de programme CSV, mais de nombreux programmes qui prennent en charge le format, y compris Excel d'ailleurs), et vous n'auriez aucun problème à ouvrir un fichier vieux de 40 ans aujourd'hui si vous en trouviez un.

Tout d'abord examinons ,le plus grand fichier de données, en utilisant les outils que nous avons appris dans la section Lire les fichiers:

$ cat 2014-01_JA.tsv

Comme pour le fichier 829-0.txt, l'ensemble des données s'affiche en cascade et il n'est pas possible de donner un sens à cette quantité de texte. Pour annuler cette concaténation en cours, ou tout autre processus du shell, appuyez sur Ctrl+C.

Dans la plupart des fichiers de données, un coup d'oil rapide sur les premières lignes nous en apprend déjà beaucoup sur la structure de l'ensemble de données, par exemple, les en-têtes des tableaux et des colonnes :

```
$ head -n 3 2014-01_JA.tsv
File    Creator Issue   Volume  Journal ISSN    ID      Citation Title  Place Labe      Language
        Publisher       Date
History_1a-rdf.tsv      Doolittle, W. E.        1       59      KIVA -ARIZONA-0023-1940
        (Uk)RN001571862         KIVA -ARIZONA- 59(1), 7-26. (1993)      A Method for Distinguishing
between Prehistoric and Recent Water and Soil Control Features  xxu     eng     ARIZONA
ARCHAEOLOGICAL AND HISTORICAL SOCIETY   1993
History_1a-rdf.tsv      Nelson, M. C.   1       59      KIVA -ARIZONA-0023-1940
        (Uk)RN001571874         KIVA -ARIZONA- 59(1), 27-48. (1993)     Classic Mimbres Land Use in the
Eastern Mimbres Region, Southwestern New Mexico         xxu     eng     ARIZONA ARCHAEOLOGICAL
```

Ensuite, nous allons nous familiariser avec un outil d'analyse de données de base : wc est la commande "word count" : elle compte le nombre de lignes, de mots et d'octets. Lançons la commande wc *.tsv pour obtenir le nombre de lignes de tous les fichiers .tsv du répertoire courant.

Les trois premières colonnes contiennent le nombre de lignes, de mots et d'octets.

Si nous n'avons qu'une poignée de fichiers à comparer, il peut être plus rapide ou plus pratique de vérifier avec Microsoft Excel, OpenRefine ou vote éditeur de texte préféré, mais lorsque nous avons des dizaines, des centaines ou des milliers de documents, la ligne de commandes Unix a un net avantage en termes de rapidité. La véritable puissance du shell réside dans la possibilité de combiner des commandes et d'automatiser des tâches. Nous y reviendrons brièvement.

Pour l'instant, nous allons voir comment construire un "pipeline" simple pour trouver le fichier le plus court en termes de nombre de lignes. Nous commençons par ajouter l'option -l pour obtenir uniquement le nombre de lignes, et non le nombre de mots et d’octets.

```
$ wc -l *.tsv
   13712 2014-01-31_JA-africa.tsv
   27392 2014-01-31_JA-america.tsv
  507732 2014-01_JA.tsv
    5375 2014-02-02_JA-britain.tsv
  554211 total
```

La commande wc elle-même n'a pas 'option pour trier la sortie, mais comme nous le verrons, nous pouvons combiner trois commandes shell différentes pour obtenir ce que nous voulons.
Tout d'abord, nous avons la commande wc -l *.tsv. Nous allons enregistrer la sortie de cette commande dans un nouveau fichier. Pour ce faire, nous redirigeons la sortie de la commande vers un fichier en utilisant le signe "plus grand que" (>), comme suit:

```
$ wc -l *.tsv > lengths.txt
```

Il n'y a pas de données affichées maintenant puisque la sortie est allée dans le fichier lengths.txt, mais nous pouvons vérifier que la sortie a bien abouti dans le

```
$ cat lengths.txt
   13712 2014-01-31_JA-africa.tsv
   27392 2014-01-31_JA-america.tsv
  507732 2014-01_JA.tsv
    5375 2014-02-02_JA-britain.tsv
  554211 total
```

fichier en utilisant cat ou less (ou Notepad ou 'importe quel éditeur de texte). Ensuite, il y a la commande sort. Nous utiliserons l'option -n pour spécifier que nous voulons un tri numérique, et non un tri lexical, nous sortirons les résultats dans un autre fichier, et nous utiliserons cat pour vérifier les résultats:

```
$ sort -n lengths.txt > sorted-lengths.txt
bash-3.2$ cat sorted-lengths.txt
    5375 2014-02-02_JA-britain.tsv
   13712 2014-01-31_JA-africa.tsv
   27392 2014-01-31_JA-america.tsv
  507732 2014-01_JA.tsv
```

Enfin, nous avons notre vieil ami head, que nous pouvons utiliser pour obtenir la première ligne du fichier sorted-lengths.txt :

```
$ head -n 1 sorted-lengths.txt
    5375 2014-02-02_JA-britain.tsv
```

Mais ce qui nous intéresse vraiment, c'est le résultat final, et non les résultats intermédiaires stockés dans lengths.txt et sorted-lengths.txt. Et si nous pouvions

envoyer les résultats de la première commande (`wc -l *.tsv`) directement à la commande suivante (`sort -n`) puis la sortie de cette commande à `head -n 1`? Heureusement, c'est possible, grâce à un concept appelé "pipes". Sur la ligne de commande, vous créé un pipe avec la barre verticale |. Essayons d'abord avec un seul :

```
$ wc -l *.tsv | sort -n
   5375 2014-02-02_JA-britain.tsv
  13712 2014-01-31_JA-africa.tsv
  27392 2014-01-31_JA-america.tsv
 507732 2014-01_JA.tsv
 554211 total
```

Remarquez qu'il s'agit exactement de la même sortie que celle qui s'est retrouvée dans notre fichier sorted-lengths.txt plus tôt. Ajoutons un autre *pipe*:

```
$ wc -l *.tsv | sort -n | head -n 1
    5375 2014-02-02_JA-britain.tsv
```

Il faut parfois un certain temps pour comprendre les pipes et les utiliser efficacement, mais il s'agit d'un 'concept très puissant que l'on retrouve non seulement dans le shell, mais aussi dans la plupart des langages de programmation.

Pipes et filtres

C'est cette idée simple qui explique le succès d'Unix. Au lieu de créer d'énormes programmes qui essaient de faire beaucoup de chose différentes, les programmers Unix se concentrent sur la création d'un grand nombre d'outils simples qui font bien chacun un travail et qui fonctionnent bien les uns avec les autres. Ce modèle de programmation est appelé "*pipes* et filtres". Nous avons déjà vu les pipes ; un filtre est un programme comme wc ou sort qui transforme un flux d'entrée en un flux de sortie. Presque tous les outils standard d'Unix peuvent fonctionner de cette manière : sauf indication contraire, ils lisent à partir de l'entrée standard, font quelque chose avec ce qu'ils ont lu et écrivent sur la sortie standard.

La clé est que tout programme qui lit des lignes de texte à partir de l'entrée standard et écrit des lignes de texte vers la sortie standard peut être combiné avec tous les autres programmes qui se comportent de la même manière. Vous pouvez et devriez écrire vos scripts de cette manière afin que vous et d'autres personnes puissent placer ces programmes dans des tuyaux pour multiplier leur puissance.

AJOUT D'UN AUTRE PIPE

Vous avez `wc -l *.tsv | sort -n | head -n 1`

- Que se passerait-il si l'on envoyait la sortie dans cat ? Essayez-le!

COMPTER LE NOMBRE DE MOTS, LES TRIER ET LES IMPRIMER

Pour compter le nombre total de lignes dans chaque fichier tsv, trier les résultats et imprimer la première ligne du fichier, nous utilisons ce qui suit:

```
wc -l *.tsv | sort -n | head -n 1
```

Changeons maintenant de scénario. Nous voulons connaître les 10 fichiers qui contiennent le plus de mots. Consultez le man wc (ou wc --help) pour voir si vous pouvez trouver l'option à utiliser pour imprimer le nombre de mots (mais pas le nombre de lignes et d'octets).
Remplissez les blancs ci-dessous pour compter les mots de chaque fichier, les mettre dans l'ordre, puis afficher les 10 fichiers contenant le plus de mots (astuce la commande sort par défaut dans l'ordre croissant).

```
wc __ *.tsv | sort __ | ___
```

COMPTER LE NOMBRE DE FICHIERS

- Prenons un autre exemple. Vous voulez savoir combien de fichiers et de répertoires se trouvent dans le répertoire actuel. Essayez de voir si vous pouvez envoyer la sortie de ls dans wc pour trouver la réponse.

ÉCRIRE DANS DES FICHIERS

- La commande date affiche la date et l'heure actuelles. Pouvez-vous écrire la date et l'heure actuelles dans un nouveau fichier appelé logfile.txt ? Vérifiez ensuite le contenu du fichier.

AJOUT À UN FICHIER

- Tandis que > écrit dans un fichier, >> ajoute quelque chose à un fichier. Essayez d'ajouter la date et l'heure actuelles au fichier logfile.txt ?

COMPTER LE NOMBRE DE MOTS

- Nous avons appris l’option -w ci-dessus, alors essayez de l'utiliser avec les fichiers .tsv.
- Si vous avez le temps, vous pouvez également essayer de trier les résultats en utilisant la commande sort. Et/ou explorer les autres options de wc.

Extraire/miner ou chercher

La recherche d'un élément dans un ou plusieurs fichiers est une opération que nous devrons souvent effectuer, c'est pourquoi nous vous présentons la commande grep (abréviation de global regular expression print, ou impression globale d'expressions régulières). Comme son nom l'indique, cette commande prend en charge les expressions régulières et n'est donc limitée que par votre imagination, la forme de vos données et, lorsque vous travaillez avec des milliers ou des millions de fichiers, la puissance de traitement dot vous disposez.

Pour commencer à utiliser grep, naviguez d'abord dans le répertoire shell_lesson, si vous n’y êtes pas. Créez ensuite un nouveau répertoire "results" :

```
$ mkdir results
```

Essayons maintenant note première recherche ou extraction:

```
$ grep 1999 *.tsv
```

Rappelez-vous que le shell développe *.tsv pour obtenir une liste de tous les fichiers .tsv du répertoire. grep recherche alors dans ces fichiers les occurrences de la chaîne "1999' et affiche les lignes correspondantes.

Chaînes de caractères

Une chaîne de caractères est une séquence de caractères, ou "une portion de texte".

Appuyez une fois sur la flèche vers le haut pour revenir à votre action la plus récente. Modifiez grep 1999 *.tsv en grep -c 1999 *.tsv et appuyez sur la touche Entrée.

```
$ grep -c 1999 *.tsv
2014-01-31_JA-africa.tsv:804
2014-01-31_JA-america.tsv:1478
2014-01_JA.tsv:28767
2014-02-02_JA-britain.tsv:284
```

L'interpréteur de commandes affiche maintenant le nombre de lignes où la chaîne 1999 est apparue dans chaque fichier. Si vous regardez la sortie de la commande précédente, cela tend à se référer au champ date de chaque article de journal.

Essayons une autre recherche :

```
$ grep -c revolution *.tsv
2014-01-31_JA-africa.tsv:20
2014-01-31_JA-america.tsv:34
2014-01_JA.tsv:867
2014-02-02_JA-britain.tsv:9
```

Nous avons obtenu le nombre d'occurrences de la chaîne revolution dans les fichiers. Maintenant, remplacez la commande ci-dessus par la commande ci-dessous et observez comment la sortie de chaque commande est différente:

```
$ grep -ci revolution *.tsv
2014-01-31_JA-africa.tsv:118
2014-01-31_JA-america.tsv:1018
2014-01_JA.tsv:9327
2014-02-02_JA-britain.tsv:122
```

C’est la même requête sauf que le décompte est insensible à la casse (y compris les occurrences de revolution et de Revolution et d'autres variantes). Notez que le décompte a été multiplié par près de 30 pour les titres 'articles de revues contenant le mot-clé "america". Comme précédemment, revenir en arrière et ajouter > results/, suivi d'un nom de fichier (idéalement au format .txt), permet d'enregistrer les résultats dans un fichier de données.

Jusqu'à présent, nous avons compté les chaînes de caractères dans les fichiers et imprimé ces chiffres dans à l’écran ou dans un fichier. Mais la véritable puissance de grep réside dans le fait que vous pouvez également l'utiliser pour créer des sous-ensembles de données tabulées (ou même de n'importe quelles données) à partir d'un ou de plusieurs fichiers.

```
$ grep -i revolution *.tsv
```

Cette commande recherche dans les fichiers définis et imprime toutes les lignes contenant revolution (sans tenir compte de la casse) à la ligne de commandes.

Laissons le shell ajouter la date du jour au nom du fichier:

```
$ grep -i revolution *.tsv > results/$(date "+%Y-%m-%d")_JAi-revolution.tsv
```

Cette opération permet d'enregistrer un autre sous-ensemble dans un nouveau fichier.

Commandes de dates alternatives

Cette façon d'écrire les dates est tellement courante que sur certaines plateformes, vous pouvez obtenir le même résultat en tapant $(date -I) au lieu de $(date "+%Y-%m-%d").

Cependant, si nous examinons ce fichier, nous constatons qu'il contient toutes les occurrences de la chaîne de caractères "revolution", y compris en tant que mot unique et en tant que partie d'autres mots tels que "revolutionary", Ce n'est peut-être pas aussi utile que nous le pensions. Heureusement, le drapeau -w demande à grep de ne rechercher que les mots entiers, ce qui nous donne une plus grande précision dans note recherche.

```
$ grep -iw revolution *.tsv > results/$(date "+%Y-%m-%d")_JAiw-revolution.tsv
```

Ce script examine les fichiers visés et exporte toutes les lignes contenant le mot revolution (sans tenir compte de la casse) vers le fichier .tsv spécifié.

```
$ wc -l results/*.tsv
  10585 results/2023-07-24_JAi-revolution.tsv
   7779 results/2023-07-24_JAiw-revolution.tsv
  18364 total
```

Nous pouvons montrer la différence entre les fichiers que nous avons créés.

L'AJOUT AUTOMATIQUE D'UN PRÉFIXE DE DATE

Remarquez que nous n'avons pas tapé ladate du jour nous-mêmes, mais que nous avons laissé la commande de date effectuer cette tâche inutile à notre place.

- Découvrez l'option "+%Y-%m-%d" et les autres options que nous aurions pu utiliser.

EXPRESSIONS RÉGULIÈRES BASE, ÉTENDUE ET COMPATIBLE PERL

Il existe malheureusement différentes manières d'écrire de expressions régulières. Dans ses différentes versions, grep prend en charge les expressions régulières "de base", au moins deux types 'expressions régulières "étendues" et "compatibles avec PERL". C'est une cause fréquente de confusion, car la plupart des tutoriels, y

compris le nôtre, enseignent les expressions régulières compatibles avec le langage de programmation PERL, mais grep utilise basic par défaut. À moins que vous ne souhaitiez vous souvenir des détails, simplifiez-vous la vie en utilisant toujours les expressions régulières les plus avancées que votre version de grep supporte (drapeau -E sur macOS X, -P sur la plupart des autres plateformes) ou lorsque vous faites quelque chose de plus complexe que la recherche d'une simple chaîne de caractères.

L'expression régulière fr[a]nc[eh] 'correspondra à "france", "french", mais aussi à "frence" et "franch". C' est généralement une bonne idée de mettre l'expression entre guillemets simples, car cela garantit que l'interpréteur de commandes l'envoie directement à grep sans aucun traitement (comme essayer d'étendre l'opérateur de caractères génériques *).

```
$ grep -iwE 'fr[ae]nc[eh]' *.tsv
```

L'interpréteur de commands imprimera chaque ligne correspondante.

Incluons l'option -o pour n'imprimer que la partie correspondante des lignes, par exemple (pratique pour isoler/vérifier les résultats :

```
$ grep -iwEo 'fr[ae]nc[eh]' *.tsv
```

RECHERCHE SENSIBLE À LA CASSE

- Recherchez toutes les occurrences sensibles à la casse d'un mot entier choisi dans les quatre fichiers .tv dérivés de ce répertoire. Imprimez vos résultats à l’écran.

- Recherchez toutes les occurrences sensibles à la casse d'un mot choisi dans les fichiers .tsv "America "et "Africa "de ce répertoire. Imprimez vos résultats à l’écran.

- Comptez toutes les occurrences sensibles à la casse d'un mot choisi dans les fichiers .tsv 'America 'et 'Africa de ce répertoire. Imprimez vos résultats à l’écran.

- Comptez toutes les occurrences insensibles à la casse de ce mot dans les fichiers .tsv "America "et "Africa "de ce répertoire. Affichez vos résultats à l’écran.

- Recherchez toutes les occurrences insensibles à la casse de ce mot dans les fichiers .tsv "America "et "Africa "de ce répertoire. Imprimez vos résultats dans le fichier results/hero.tsv.

- Recherchez toutes les occurrences insensible à la casse de ce mot entier dans les fichiers .tsv "America "et "Africa "de ce répertoire. Imprimez vos résultats dans le fichier results/hero-i.tsv.

- Utilisez des expressions régulières pour trouver tous les numéros ISSN (quatre chiffres suivis d'un trait 'union suivi de quatre chiffres) dans 2014-01 JA.tsv et imprimez les résultats dans un fichier results/issns.tsv. Notez que vous devrez peut-être utiliser le drapeau -E (ou -P avec certaines versions de grep, par exemple avec Git Bash sur Windows).

RECHERCHE DE VALEURS UNIQUES

Si vous envoyez quelque chose à la commande uniq, celle-ci filtrera les lignes dupliquées adjacentes. Cependant, pour que la commande uniq ne renvoie que des valeurs uniques, elle doit être utilisée avec la commande sort.

- Essayez d'acheminer la sortie de la commande du dernier exercice vers le tri, puis d'acheminer ces résultats vers uniq et wc -l pour compter le nombre de valeurs ISSN uniques.

Utilisation d'une boucle pour compter les mots

Nous allons maintenant utiliser une boucle pour automatiser le comptage de certains mots dans un document. Pour ce faire, nous utiliserons le livre électronique Little Women du Projet Gutenberg. Le fichier se trouve dans le dossier shell_lesson et s'appelle pg514.txt. Renommons le fichier en littlewomen. txt.

```
$ mv pg514.txt littlewomen.txt
```

Cela permet de renommer le fichier en quelque chose de plus facile à taper.

Créons maintenant notre boucle. Dans cette boucle, nous demanderons à l'ordinateur de parcourir le texte à la recherche du nom de chaque fille et de

```
$ for name in "Jo" "Beth" "Meg" "Amy"
> do
>   echo "$name"
>   grep -wo "$name" littlewomen.txt |wc -l
> done
Jo
   1355
Beth
    459
Meg
    683
```

compter le nombre de fois qu'il apparaît. Les résultats seront imprimés à l'écran.

Que se passe-t-il dans la boucle?

`echo "$name"` affiche la valeur actuelle de $name
`grep "$name" littlewomen.txt` recherche chaque ligne contenant la valeur stockée dans $name. L'option -w ne trouve que le mot entier qui correspond à la valeur stockée dans $name et l'option -o extrait cette valeur de la ligne dans laquelle elle se trouve pour vous donner les mots réels à compter comme des lignes en soi.

La sortie de la commande grep est redirigée par le pipe, | (sans le pipe et le reste de la ligne, la sortie de grep s'imprimerait directement à l'écran). wc -l compte le nombre de lignes envoyées par grep. Comme grep ne renvoie que les lignes contenant la valeur stockée dans $name, wc -l correspond au nombre d'occurrences du nom de chaque fille.

POURQUOI LES VARIABLES ONT-ELLES DES GUILLEMETS DOUBLES?

Dans la boucle, nous avons appris à utiliser "$.." pour éviter que les spaces blancs ne soient mal interprétés. Pourquoi pourrions-nous omettre les "-quotes" dans l'exemple ci-dessus?

Que se passe-t-il si vous ajoutez "Louisa May Alcott" à la première ligne de la boucle et que vous supprimez les guillemets entourant $name dans le code de la boucle?

SÉLECTION DE COLONNES DANS NOTRE ENSEMBLE DE DONNÉES D'ARTICLES

Lorsque vous recevez des données, elles contiennent souvent plus de colones ou de variables que vous 'en avez besoin pour votre travail. Si vous souhaitez sélectionner uniquement les colonnes dont vous avez besoin pour votre analyse, vous pouvez utiliser la commande cut. cut est un outil permettant d'extraire des sections d'un fichier. Par exemple, supposons que nous voulions conserver uniquement les colonnes Créateur, Volume, Revue et Citation de nos données d'articles. Avec la commande cut, nous pourrions:

```
$ cut -f 2,4,5,8 2014-01_JA.tsv | head -n 5
Creator Volume Journal Citation
Doolittle, W. E. 59 KIVA -ARIZONA-KIVA -ARIZONA- 59(1), 7-26. (1993)
Nelson, M. C. 59 KIVA -ARIZONA-KIVA -ARIZONA- 59(1), 27-48. (1993)
Deegan, A. C. 59 KIVA -ARIZONA-KIVA -ARIZONA- 59(1), 49-64. (1993)
Stone, T. 59 KIVA -ARIZONA-KIVA -ARIZONA- 59(1), 65-82. (1993)
```

Ci-dessus, nous avons utilisé cut et l'option -f pour indiquer les colonnes à conserver, cut fonctionne par défaut avec des fichiers délimités par des tabulations. Nous pouvons utiliser l'option -d pour replacer la tabulation par une virgule, un point-virgule ou un autre délimiteur. Si vous n'êtes pas sûr de la position de vos colonnes et que le fichier contient des en-têtes sur la première ligne, vous pouvez utiliser head -n 1 <fichier> pour les imprimer.

- Sélectionnez les colonnes Numéro, Volume, Langue, Éditeur et dirigez le résultat dans un nouveau fichier. Vous pouvez le nommer quelque chose comme 2014-01_JA_ivlp.tsv.

Avec cut, vous ne pouvez pas extraire les colonnes dans le désordre, mais vous pouvez utiliser la commande awk pour ce faire.

```
$ awk -F '\t' '{ print $5 "\t" $2 "\t" $4 "\t" $8 }' 2014-01_JA.tsv| head -n 5
```

ou

```
$ awk 'BEGIN {FS = "\t" ; OFS = "\t"} {print $5 $2 $8 $4}' 2014-01_JA.tsv |head -n 5
```

Dans la deuxième ligne, vous définissez une fois le séparateur de champs de sortie (OFS). Pour en savoir plus sur awk, cliquez ici:

https://www.gnu.org/software/gawk/manual/html_node/index.html

POINTS CLÉS

L'interpréteur de commandes put être utilisé pour compter les éléments des documents
L'interpréteur de commandes put être utilisé pour rechercher des motifs dans les fichiers
Les commandes peuvent être utilisées pour compter et exploiter 'importe quel nombre de fichiers
Les commandes et les drapeaux peuvent être combinés pour construire des requêtes complexes spécifiques à vote travail.

6. Travailler avec du texte libre

Jusqu'à présent, nous avons vu comment utiliser la ligne de commandes Unix pour manipuler, compter et exploiter des données tabulées. Certaines données de bibliothèque, en particulier les documents numérisés, sont beaucoup plus désordonnées que les métadonnées tabulaires. Néanmoins, un grand nombre de ces techniques peuvent être appliqués à des données non tabulées telles que du texte libre. Nous devons réfléchir attentivement à ce que nous comptons et à la manière dont nous pouvons tirer le meilleur parti du shell Unix.

Heureusement, il y a beaucoup de gens qui font ce genre de travail et nous pouvons emprunter ce qu'ils font comme introduction au travail avec ces fichiers plus complexes. Pour ce dernier exercice, nous allons donc faire un petit bond en avant en termes de difficulté, dans un scénario où nous n'apprendrons pas tout ce qui se passe en détail et où nous ne discuterons pas longuement de chaque commande. Nous allons préparer et découper des texts pour démontrer certaines des applications potentielles de la ligne de commandes. Et lorsque les commandes que nous avons apprises sont utilisées, j'ai laissé une partie du travail à faire à votre charge - alors n'hésitez pas à vous référer à vos notes si vous êtes bloqués!

Option 1: texte transcrit à la main

Choisir un texte et le nettoyer
Nous allons travailler avec le fichier gulliver.txt, que nous avons créé dans la section 3, "Travailler avec des fichiers et des répertoires". Vous devriez (toujours) travailler dans le répertoire shell_ lesson.

```
bash-3.2$ less -N gulliver.txt

     1 The Project Gutenberg eBook, Gulliver's Travels, by Jonathan Swift
     2
     3
     4 This eBook is for the use of anyone anywhere at no cost and with
     5 almost no restrictions whatsoever.  You may copy it, give it away or
     6 re-use it under the terms of the Project Gutenberg License included
     7 with this eBook or online at www.gutenberg.org
     8
     9
    10
    11
    12
    13 Title: Gulliver's Travels
    14        into several remote nations of the world
    15
    16
    17 Author: Jonathan Swift
    18
    19
    20
    21 Release Date: June 15, 2009  [eBook #829]
    22
```

Examinons le fichier.
Nous allons commencer par utiliser la commande sed. Cette commande permet d'éditer directement les fichiers.

```
$ sed '9352,9714d' gulliver.txt > gulliver-nofoot.txt
```

La commande sed, associée à la valeur d, examine le fichier gulliver.txt et supprime toutes les valeurs situées entre les lignes spécifiées. L'action > invite alors le script à insérer le texte modifié dans le nouveau fichier spécifié.

```
$ sed '1,37d' gulliver-nofoot.txt > gulliver-noheadfoot.txt
```

Cette opération est la même que la précédente, mais pour l'en-tête.

Vous disposez à présent d'un texte plus propre. L'étape suivante consiste à le préparer davantage à une analyse rigoureuse.

Nous allons maintenant utiliser la commande tr, qui permet de transposer ou de supprimer des caractères. Tapez et exécutez

```
$ tr -d '[:punct:]\r' < gulliver-noheadfoot.txt > gulliver-noheadfootpunct.txt
```

Cette opération utilise la commande tr et une syntaxe spéciale pour supprimer toute la ponctuation ([:punct :]) et les retours à la ligne (\r). Elle nécessite également l'utilisation de la redirection de sortie > que nous avons vue et de la redirection d'entrée < que nous n'avons pas vue.

Enfin, régularisez le texte en supprimant toutes les lettres majuscules.

```
$ tr '[:upper:]' '[:lower:]' < gulliver-noheadfootpunct.txt > gulliver-clean.txt
```

Ouvrez le fichier gulliver-clean. txt dans un éditeur de texte. Notez comment le texte a été transformé pour être prêt à être analysé.

PULLING A TEXT APART, COUNTING WORD FREQUENCIES

Nous sommes maintenant prêts à décortiquer le texte.

```
$ tr ' ' '\n' < gulliver-clean.txt | sort | uniq -c | sort -nr > gulliver-final.txt
```

Ici, nous avons fait un usage étendu des pipes que nous avons vus dans la section 5. La première partie de ce script utilise à nouveau la commande tr, cette fois-ci pour traduire chaque espace vide en \n qui s'affiche comme une nouvelle ligne. Chaque mot du fichier aura à ce stade sa propre ligne.

La deuxième partie utilise la commande de tri pour réorganiser le texte de son ordre initial en une configuration alphabétique.

La troisième partie utilise uniq, une autre nouvelle commande, en combinaison avec 'option -c pour supprimer les lignes dupliquées et produire un décompte des mots de ces duplications.

La quatrième et dernière partie trie à nouveau le texte en fonction du nombre de comptes générés lors de la troisième étape.

Nous avons maintenant décortiqué le texte et produit un décompte pour chaque mot qu'il contient. Il s'agit de données que nous pouvons manipuler et visualiser, qui peuvent constituer la base de nos recherches et que nous pouvons comparer à d'autres textes traités de la même manière. Et si nous avons besoin d'exécuter un autre ensemble de transformations pour une analyse différente, nous pouvons revenir à gulliver-clean.txt pour commencer ce travail.

Et tout cela en utilisant quelques commandes sur une ligne de commande sans prétention mais très puissante.

Option 2: Récupérer le texte d'une page web et le nettoyer

Nous travaillerons avec le fichier diary.html.

Examinons le fichier.

```
$ less -N diary.html
     1 <!-- This document was created with HomeSite v2.5 -->
     2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
     3 <html>
     4
     5 <head>
     6
     7
     8 <script type="text/javascript" src="/static/js/analytics.js"></script>
     9 <script type="text/javascript">archive_analytics.values.server_name="www
```

Nous allons commencer par utiliser la commande sed. Cette commande permet d'éditer directement les fichiers.

```
$ sed '265,330d' diary.html > diary-nofoot.txt
```

La commande sed, associée à la valeur d, examine le fichier diary. html et supprime
toutes les valeurs entre les lignes spécifiées. L'action > invite alors le script à insérer le texte modifié dans le nouveau fichier spécifié.

```
$ sed '1,221d' diary-nofoot.txt > diary-noheadfoot.txt
```

Cette opération est la même que la précédente, mais pour l'en-tête.
Vous disposez à présent d'un texte plus propre. L'étape suivante consiste à le préparer davantage à une analyse rigoureuse.

Tout d'abord, nous allons supprimer toutes les balises html. Tapez et exécutez:

```
$ sed -e 's/<[^>]*>//g' diary-noheadfoot.txt > diary-notags.txt
```

Ici, nous utilisons une expression régulière (voir la Leçon sur les expressions régulières) pour trouver toutes les balises html valides (tout ce qui se trouve entre crochets) et les supprimer. Il s'agit d'une expression régulière complexe, alors ne vous inquiétez pas trop de son fonctionnement ! Le script nécessite également l'utilisation de la redirection de sortie > que nous avons vue et de la redirection 'entrée < que nous n'avons pas vue.

Nous allons commencer par utiliser la commande tr, utilisée pour traduire ou supprimer des caractères. Tapez et exécutez:

```
$ tr -d '[:punct:]\r' < diary-notags.txt > diary-notagspunct.txt
```

Cette commande utilise la commande translate et une syntaxe spéciale pour supprimer toutes les ponctuations ([: punct :]) et les retours chariot (\r).

Enfin, régularisez le texte en supprimant toutes les lettres majuscules.

```
$ tr '[:upper:]' '[:lower:]' < diary-notagspunct.txt > diary-clean.txt
```

Ouvrez le fichier diary-clean.txt dans un éditeur de texte. Notez comment le texte a été transformé pour être prêt à être analysé.

Nous sommes maintenant prêts à décortiquer le texte.

```
$ tr ' ' '\n' < diary-clean.txt | sort | uniq -c | sort -nr > diary-final.txt
```

Ici, nous avons fait un usage étendu des tuyaux que nous avons vus dans la section 5.
La première partie de ce script utilise à nouveau la commande tr, cette fois pour traduire chaque espace vide en \n qui s'affiche comme une nouvelle ligne. Chaque mot du fichier aura à ce stade sa propre ligne.

La deuxième partie utilise la commande de tri pour réorganiser le texte de son ordre initial en une configuration alphabétique.

La troisième partie utilise uniq, une autre nouvelle commande, en combinaison avec l'option -c pour supprimer les lignes dupliquées et produire un décompte des mots de ces duplications.

La quatrième et dernière partie trie à nouveau le texte en fonction du nombre de doublons générés lors de la troisième étape.

Nous avons maintenant décortiqué le texte et produit n décompte pour chaque mot qu'il contient. Il s'agit de données que nous pouvons manipuler et visualiser, qui peuvent constituer la base de nos recherches et que nous pouvons comparer à d'autres textes traités de la même manière. Et si nous devons effectuer un autre ensemble de transformations pour une autre analyse, nous pouvons revenir au fichier diary-final. txt pour commencer ce travail.

Et, encore une fois, tout cela en utilisant quelques commandes sur une ligne de commande sans prétention mais très puissante.

Informations supplémentaires

The best AWK one-liners
https://www.shebanglinux.com/best-awk-one-liners/

Learning AWK
https://www.tutorialspoint.com/awk/awk_workflow.htm

Regular expressions info
https://www.regular-expressions.info/posixbrackets.html

Information about SED and AWK (not recent but a lot of details)
https://tldp.org/LDP/abs/html/sedawk.html

Regular expressions info

https://www.regular-expressions.info/posixbrackets.html

A fun way of learning regular expressions
https://regex101.com/